UNIT II

2 Requirements Analysis and Specification

Syllabus

Requirement analysis and specification - Requirements gathering and analysis - Software Requirement Specification - Formal system specification - Finite State Machines - Petrinets - Object modelling using UML - Use case Model - Class diagrams - Interaction diagrams - Activity diagrams - State chart diagrams - Functional modelling - Data Flow Diagram - CASE TOOLS.

Contents

2.1 Introduction to Software Requirements

2.2 Functional and Non Functional Requirements ..... **Dec.-13,17,22, May-13,** ..... Marks 11

2.3 Requirements Engineering Process..... **May-15,17, Dec.-15,19,** ..... Marks 16

2.4 Feasibility Studies ..... **May-17,22, Dec.-17,** ..... Marks 8

2.5 Requirement Gathering and Analysis ..... **Dec.-16,22,** ..... **May-08,17,18,19,** ..... Marks 16

2.6 Software Requirement Specification..... **Dec.-11,19, 20,** ..... **May-13,14,16,19,22,** ..... Marks 16

2.7 Formal System Specification

2.8 Finite State Machines

2.9 Petri Nets ..... **Dec.-19,** ..... Marks 13

2.10 Object Modeling using UML..... **May-11,14, Dec.-18,** ..... Marks 16

2.11 Use Case Model ..... **May-11,12,13,14,18,19,** ..... **Dec.-11,12,13,15,16,17,18,**..... Marks 16

2.12 Class Diagrams ..... **May-11,12,16,17,18,19,** ..... **Dec.-11,12,13,14,15,16,17,18,**... Marks 16

2.13	Interaction Diagrams	May-11,14,18,19,	
		Dec.-13,14,15,16,17,18,	Marks 16
2.14	Activity Diagrams	Dec.-13,18,	Marks 16
2.15	State Chart Diagrams	Dec.-11,13,14,	
		May-11,14,19,	Marks 16
2.16	Functional Modelling	May-16,	Marks 8
2.17	Data Flow Diagram	May-08,10,15,17,18,19,	
		Dec.-13,15,16,17,	Marks 16
2.18	Two Marks Questions with Answers		

2.1 Introduction to Software Requirements

What is requirement engineering ?

Requirement engineering is the process of

- establishing the services that the customer requires from a system
- and the constraints under which it operates and is developed.

The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

What is a requirement ?

A requirement can range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

The requirement must be open to interpretation and it must be defined in detail.

Types of requirements

The requirements can be classified as

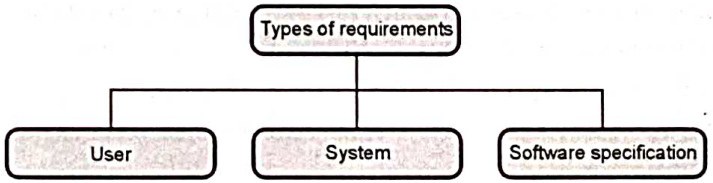


Fig. 2.1.1 Types of requirements

- **User requirements**

It is a collection of statements in natural language plus description of the services the system provides and its operational constraints. It is written for customers.

- **System requirements**

It is a structured document that gives the detailed description of the system services. It is written as a contract between client and contractor.

- **Software specification**

It is a detailed software description that can serve as a basis for design or implementation. Typically it is written for software developers.

2.2 Functional and Non Functional Requirements

AU : Dec.-13,17,22, May-13, Marks 11

Software system requirements can be classified as functional and non functional requirements.

2.2.1 Functional Requirements

- Functional requirements should describe all the required functionality or system services.
- The customer should provide statement of service. It should be clear how the system should react to particular inputs and how a particular system should behave in particular situation.
- Functional requirements are heavily dependant upon the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.
- For example : Consider a library system in which there is a single interface provided to multiple databases. These databases are collection of articles from different libraries. A user can search for, download and print these articles for a personal study.

From this example we can obtain functional requirements as -

1. The user shall be able to search either all of the initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in the document store.
3. A unique identifier (ORDER_ID) should be allocated to every order. This identifier can be copied by the user to the account's permanent storage area.

2.2.1.1 Problems Associated with Requirements

- **Requirements imprecision**

1. Problems arise when requirements are not precisely stated.
2. Ambiguous requirements may be interpreted in different ways by developers and users.
3. Consider meaning of term 'appropriate viewers'.

- **User intention** - Special purpose viewer for each different document type;
- **Developer interpretation** - Provide a text viewer that shows the contents of the document.

- **Requirements completeness and consistency -**

The requirements should be both complete and consistent. *Complete* means they should include descriptions of all facilities required. *Consistent* means there should be no conflicts or contradictions in the descriptions of the system facilities.

Actually in practice, it is impossible to produce a complete and consistent requirements document.

2.2.2 Non Functional Requirements

- The non functional requirements define system properties and constraints.
- Various properties of a system can be : Reliability, response time, storage requirements. And constraints of the system can be : Input and output device capability, system representations etc.
- Process requirements may also specify programming language or development method.
- Non functional requirements are more critical than functional requirements. If the non functional requirements do not meet then the complete system is of no use.

2.2.2.1 Types of Non Functional Requirements

The classification of non functional requirements is as given below. (See Fig. 2.2.1 on next page)

Product requirements

These requirements specify how a delivered product should behave in a particular way. For instance: execution speed, reliability.

Organizational requirements

The requirements which are consequences of organizational policies and procedures come under this category. For instance: process standards used implementation requirements.

External requirements

These requirements arise due to the factors that are external to the system and its development process. For instance : interoperability requirements, legislative requirements.

In short, non functional requirements arise through

i) User needs
ii) Because of budget constraints
iii) Organizational policies
iv) The need for interoperability with other software or hardware systems
v) Because of external factors such as safety regulations.

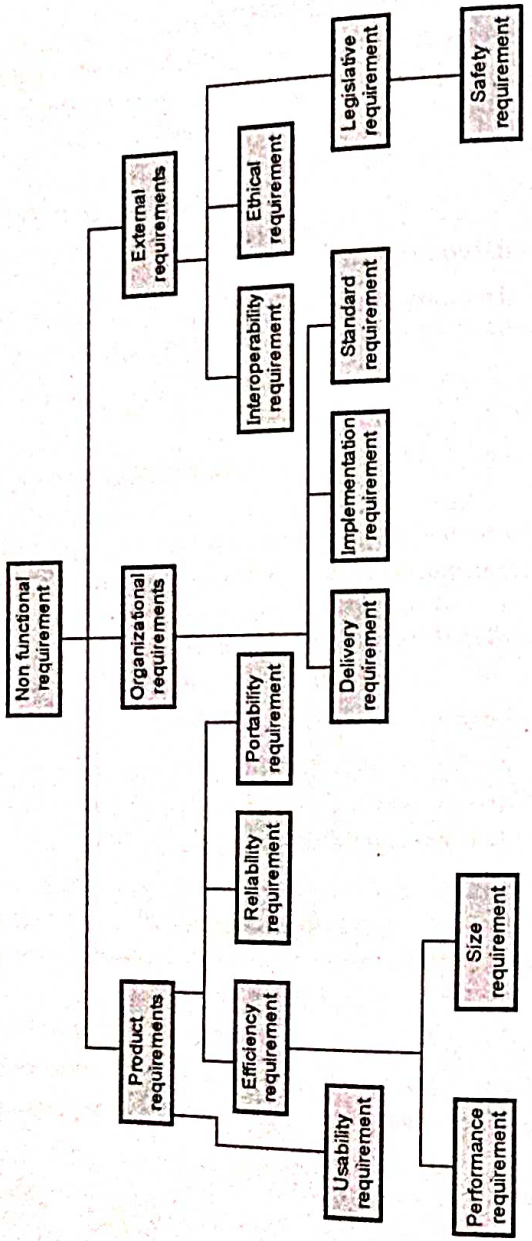


Fig. 2.2.1 Types of non functional requirement

Metrics used for specifying the non functional requirements

Property	Metric
Speed	Events per response time processed transactions per second.
Size	Kilo bytes.
Reliability	Mean time to failure. Rate of failure. Occurrence availability.
Robustness	Time to restart after failure. Probability of events causing failure.
Portability	Number of target statements.

2.2.2.2 Domain Requirements

- Domain requirements are derived from the application domain of the system instead of specific user needs.
- These requirements make use of domain terminologies specific to the existing domain concept.
- The domain requirements may be in the form of new functional requirements, constraints on existing functional requirement or guidance on how to carry out certain computation.
- These are the specialised requirements and hence software engineers find it difficult to co-relate the domain requirements with the system requirements.
- It is important to specify the domain requirements otherwise the system will not work properly.
- **Example : Domain requirements for the library system.**
- There should be user interface for handling the databases. These interfaces should be according to some international standard.
- If there is copyright restriction on some document then it should get printed locally on the server. The copies of such document should not get created.

2.2.3 Difference between Functional and Non Functional Requirements

Functional requirements	Non functional requirements
The functional requirements specify the features of the software system	The non functional requirements specify the properties of the software system.

Functional requirements describe what the product must do	Non functional requirements describe how the product should perform.
The functional requirements specify the actions with which the work is concerned.	The non functional requirements specify the experience of the user while using the system.
Example : For a library management system, allowing user to read the article online is a functional requirement.	Example : For a library management system, for a user who whishes to read the article online must be authenticated first.

Metrics used for specifying Non Functional Requirements

The metrics used for specifying the non functional requirements are -

1. **Reliability :** The application should be highly reliable and it should generate all the updated information in correct order.
2. **Availability :** Any information must be quickly available from computer to the authorized user.
3. **Security :** The application must be password protected. This feature is essential to avoid any unauthorized access to the application.
4. **Maintainability :** The application should be maintained in such a manner that if any change in requirement occurs then it should be easily incorporated in an individual module.
5. **Extensibility :** The application should be maintained in such a manner that if any new requirement occurs then it should be easily incorporated in an individual module.
6. **Portability :** The application should be portable on the desired operating system.
7. **Reusability :** The application should have the ability that a segment of source code can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time.
8. **Resource utilization :** The application should make use of resources to its maximum capability.

Example 2.2.1 *For the requirement given below, identify stakeholders, functional and non-functional requirements*

A software is to be built that will control an Automated Teller Machine (ATM). The ATM machine services customers 24 × 7. ATM has a magnetic stripe reader for reading an ATM card, a keyboard and display for interaction with the customer, a slot for depositing envelopes, a dispenser for cash, a printer for printing receipts and a switch that allows an operator to start/stop a machine.

The ATM services one customer at a time. When a customer inserts an ATM card and enters the Personal Identification Number (PIN), the details are validated for each transaction. A customer can perform one or more transactions. Transactions made against each account are recorded so as to ensure validity of transactions.

If PIN is invalid, customer is required to re-enter PIN before making a transaction. If customer is unable to successfully enter PIN after three tries, card is retained by machine and customer has to contact bank.

The ATM provides the following services to the customer :

1) Withdraw cash in multiples of 100

2) Deposit cash in multiples of 100

3) Transfer amount between any two accounts.

4) Make balance enquiry.

5) Print receipt.

Each of the above transactions must be made within 60 seconds in case the time exceeds 60 seconds, then the transaction is cancelled automatically. Also, if the machine is not used for more than two minutes after entry of card, the card is retained by the machine. An operator panel with a key-operated switch allows an operator to start and stop the servicing of customers. When the switch is moved to the "off position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc. The operator is required to verify and enter the total cash on hand before starting the system from this panel. **AU : Dec.-13, Marks 11**

Solution : i) Stakeholders

1. Bank customer
2. Service operator
3. Hardware and software maintenance engineers
4. Database administrators
5. Banking regulators
6. Security administrator

ii) Functional requirements

1. There should be the facility for the Customer to insert a card.
2. The system should first validate card and PIN.
3. The system should allow the customer to deposit amount in the bank.
4. The system should dispense the cash on withdrawal.
5. The system should provide the printout for the transaction.
6. The system should make the record of the transactions made by particular customer.
7. On invalid PIN entry for three times the card should be retained by the system.
8. The cash withdrawal is allowed in multiple of 100.
9. The cash deposition is allowed in multiple of 100.
10. The customer is allowed to transfer amount between the two accounts.
11. The customer is allowed to know the balance enquiry.
12. The customer is allowed to get the printout for desired transaction.
13. The system should be efficient.

iii) Non functional requirements

1. Each of the transaction should be made within 60 seconds. If the time limit is exceeded ,then cancel the transaction automatically.
2. If there is no response from the bank computer after request is made within the minutes then the card is rejected with error message.
3. The bank dispenses money only after the processing of withdrawal from the bank. That means if sufficient fund is available in user's account then only the withdrawal request is processed.
4. Each bank should process the transactions from several ATM centers at the same time.
5. The machine should be loaded with sufficient fund in it.

Example 2.2.2 *List the stakeholders and all types of requirements for an online train reservation system.* **AU : Dec.-17, Marks 7**

Solution :

1) Passenger
2) Database Administrator
3) Booking Clerk
4) Bank

Functional requirements

1) The online booking made by the customer should be associated with the account.
2) Booking confirmation must be sent to the user on specified contact details.
3) The system should provide a search option to the user so that user can search for required train and for required number of reservations.
4) The system should allow the user to make online payments.
5) The system should allow user to book for tickets.
6) The system should allow the user to cancel the booking and half of the amount paid by the customer must be refunded to him.

Non functional requirements

1) The system must be available to the user for 24*7.
2) The system should accept the payments via different payment methods like PayPal, Wallets, Cards, vouchers etc.
3) Use of captcha and encryption to avoid bots from booking tickets.
4) The response time of the system while searching, booking or cancellation operation should be very less.

Example 2.2.3 *Illustrate the basic issues in non-functional requirement and how it is more critical than functional requirements.* **AU : Dec.-22, Marks 6**

Solution : Basic issues : Refer to Metrics used for non-functional requirements from section 2.2.3

In many cases, the non-functional requirements are more critical than the functional requirements. The non-functional requirements define the success and acceptance of the system. Users may compromise on some functional requirements but they can not compromise system performance, reliability, security and user-friendliness. Failure to these requirements leads to more operational costs and even business failure.

Review Question

1. *Explain the metrics used for specifying non functional requirements.* **AU : May-13, Marks 8**

2.3 Requirements Engineering Process

AU : May-15, 17, Dec.-15,19, Marks 16

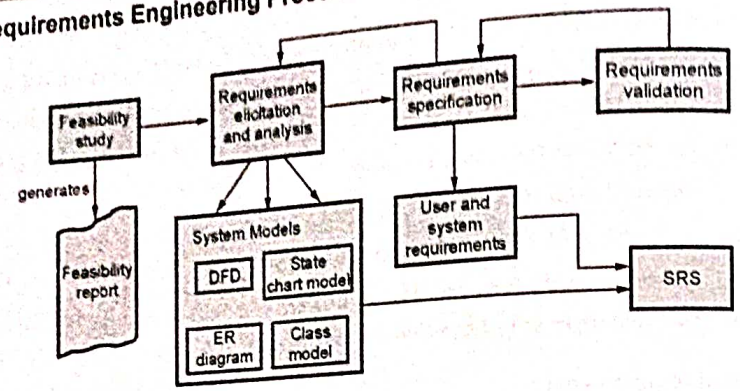


Fig. 2.3.1 Requirement engineering process

- A Requirement Engineering is a process in which various activities such as **discovery, analysis and validation** of system requirements are done.

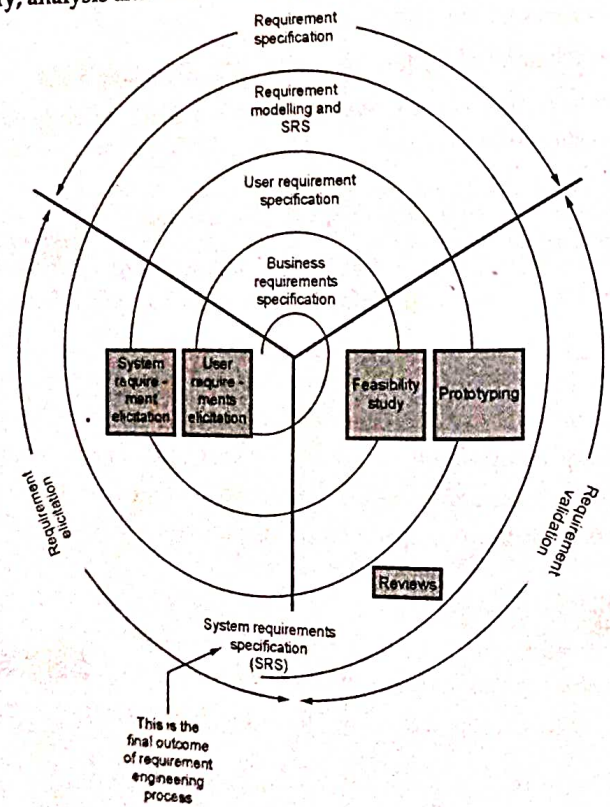


Fig. 2.3.2 Spiral model of requirements engineering process

- It begins with feasibility study of the system and ends up with requirement validation. Finally the requirement document has to be prepared.
- This process is a three stage activity where the activities are arranged in the iterative manner. In the early stage of this process most of the time is spent on understanding the system by understanding the high-level non functional requirements and user requirements. The spiral model of requirement engineering process is as shown in Fig. 2.3.2.

Requirement engineering process performs following **seven distinct functions -**

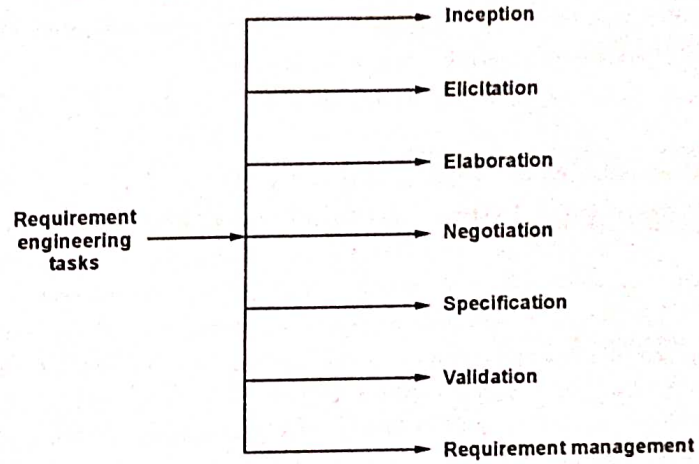


Fig. 2.3.3 Requirement engineering tasks

Let us now discuss these tasks in detail -

Inception : The inception means specifying the **beginning** of the software project. Most of the software projects get started due to business requirements. There may be potential demand from the market for a particular product.

The purpose of inception is to -

1. Establish the basic understanding of the project.
2. Find out all possible solutions and to identify the nature of the solution.
3. Establish an effective communication between developer and the customer.

Elicitation : Requirements elicitation means **requirements discovery.** Requirements elicitation is very difficult task.

Following are the reasons for : ***why it is difficult to understand customer wants ? -***

1. Customer sometimes is unable to specify the **scope of the project.** Sometimes customers specify too many technical details and this may increase the confusion.

2. There is difficulty in **understanding the problem.** Sometimes customer could not decide what are their needs and wants. Sometimes they have got poor understanding of capabilities and limitations of the existing computing environment.

   Sometimes customers find it **difficult to communicate** with the system engineer about their needs. Sometimes customers may have got **some conflicting requirements.** This ultimately results in specifying **ambiguous requirements.**

3. As project progresses the needs or requirements of the customers changes. This creates a problem of volatility.

To overcome these problems the requirements gathering must be done very systematically.

Elaboration : Elaboration is an activity in which the information about the requirements is **expanded** and **refined**. The **goal** of elaboration activity is to prepare a technical model of software functions, features and constraints. The elaboration consists of several **modelling and refinement tasks.** In this process several **user scenarios** are created and refined. During elaboration, each user scenario is parsed and various **classes** are identified. These classes are nothing but the business entities that are visible to end user. Then the **attributes** and **services** (functions) of these classes are defined. Then the relationship among these classes is identified. Finally the **analysis model** gets developed during the elaboration phase.

Negotiation : Sometimes customer may **demand for more** than that is achieved or there are certain situations in which customer demands for something which cannot be achieved in **limited business resources.** To handle such situations requirement engineers must convince the customers or end users by solving various **conflicts.**

Specification : A specification can be a **written document, mathematical** or **graphical model, collection of use case scenarios** or may be the **prototypes.** There is a need to develop a **standard specification** in which requirements are presented in consistent and understandable manner.

Validation : Requirement validation is an activity in which requirement specification is analyzed in order to ensure that the requirements are specified unambiguously. If any inconsistencies, omissions and errors are identified then those are corrected or modified during the validation.

Requirement management : Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

Example 2.3.1 *What does win-win mean in the context of negotiation during requirements engineering activity ?* **AU : Dec.-19, Marks 4**

Solution : From customer point of view the win situation is when the customer gets satisfied when most of its requirements or expectations from the system gets satisfied. From developers point of view the win situation is when the system can be developed within the given budget and within the given schedule.

Review Questions

1. *What is requirements engineering ? Explain in detail the various processes in requirements engineering.* **AU : May-17, Marks 13**
2. *Write about the following requirements engineering activities.*
   (i) Inception (ii) Elicitation (iii) Elaboration (iv) Negotiation (v) Specification
   (vi) Validation (vii) Requirements management **AU : May-15, Marks (2+3+3+2+2+2+2)**
3. *Explain the software requirement engineering process with neat diagram.* **AU : Dec.-15, May-17, Marks 16**
4. *Discuss the distinct tasks involved in requirement engineering process.* **AU : Dec.-19, Marks 9**

2.4 Feasibility Studies

AU : May-17,22, Dec.-17, Marks 8

Definition : *A feasibility study is a study made to decide whether or not the proposed system is worthwhile.*

The **focus** of feasibility study is to check

- If the system contributes to **organizational objectives.**
- If the system can be engineered using **current technology.**
- If the system is within the given **budget.**
- If the system can be **integrated** with other useful systems.

The implementation of feasibility study is based on the information assessment (what is required), information collection and report writing.

While performing the feasibility study, following questionnaires to the people in the organization should be asked -

1. What if the system wasn't implemented ?
2. What are current process problems ?
3. How will the proposed system help ?
4. What will be the integration problems ?
5. Is new technology needed? With what skills ?
6. What facilities must be supported by the proposed system ?

Feasibility study should be done with the help of project managers who is going to handle that particular project, software engineers who are about to develop that system, technical experts and customers who will be using the system. Typically the feasibility study should be completed within **two to three weeks.**

Finally the **feasibility study report** has to be prepared.

Types of Feasibility Study

1. **Technical Feasibility :** It is the measure of technical resources such as hardware components, software techniques and skilled persons.
2. **Economical Feasibility :** It is the measure whether finances or funds are available for proposed system.
3. **Operational Feasibility :** It is a measure of how well the solution of problems or a specific alternative solution will work in the organization.
4. **Schedule Feasibility :** It is the establishment of time limit for completion of the project. This kind of feasibility is dependent upon available manpower and economical support for the project.

Thus the feasibility study helps in understanding the requirements of the system. Hence feasibility study affects the requirement collection implicitly or explicitly.

Review Questions

1. *Explain the feasibility studies. What are the outcomes ? Does it have implicit or explicit effects on software requirement collection ?* **AU : May-17, Marks 8**
2. *What is feasibility study ? How it helps in requirement engineering process ?* **AU : Dec.-17, Marks 3**
3. *What is feasibility study ? What are the feasibilities to be analyzed in software development ?* **AU : May 22, Marks 6**

2.5 Requirement Gathering and Analysis

AU : Dec.-16,22, May-08,17,18,19, Marks 16

After performing feasibility study the requirements elicitation and analysis can be done. Requirement **elicitation means discovery** of all possible requirements. After identifying all possible requirements the analysis on these requirements can be done. Software engineers communicate the end-users or customers in order to find out certain information such as: application domain, expected services from the system, the expected performance level of the system. From this information even constraints of the system can be decided.

2.5.1 Stakeholders

This is a commonly used term in software engineering. The stakeholder means the person(s) who will be affected by the system. For example : end-user, system maintenance engineers or software engineers can be stakeholders. Following is the list of **problems** encountered in understanding the requirements of the system.

Problems	Description
Unrealistic expectations	Stakeholders sometimes unable to specify what they want exactly. Sometimes they specify unrealistic demands.
Differences in the requirements	Different stakeholders specify different requirements. The requirement engineer has to resolve the conflicts in the requirements with proper communication with the stakeholders
Economic and business environment	The economic and business environment is dynamic due to which there may be change in the requirements or change in the stakeholders. Both these things may affect the requirement analysis process.
Political changes	Sometimes political factors affect heavily on the need for the system. Hence there may be change in the requirements or stakeholders which ultimately affect the requirement elicitation and analysis.

2.5.2 Requirement Elicitation and Analysis Process

The spiral model shown in Fig. 2.5.1 depicts the requirement elicitation and analysis process. (See Fig. 2.5.1 on next page)

The process activities are -

1. Requirement discovery : By having effective communication with the customers the requirements can be identified.

2. Requirements classification and discovery : All the unstructured requirements can be categorised systematically depending upon their nature. And they are arranged in groups.

3. Requirement prioritisation : There are some conflicting requirements. Hence the requirements are prioritised first. If there are some unrealistic requirements then negotiations are made and only realistic prioritised requirements are collected. If any conflict occurs then it its resolved by requirement engineers.

4. Requirement documentation : This is the specification of all the requirements. And an important requirement document is created.

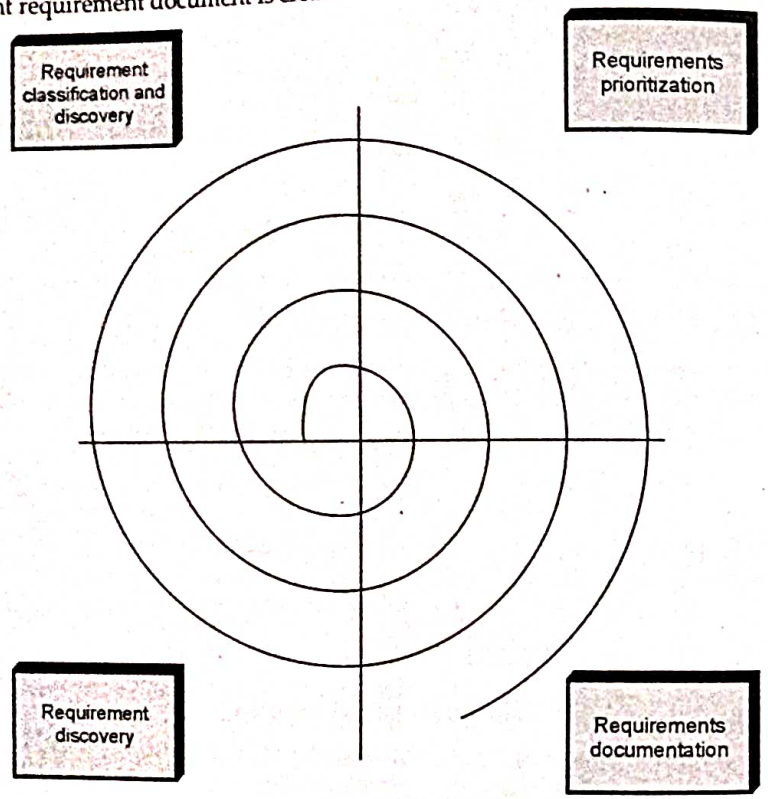


Fig. 2.5.1 Requirements elicitation and analysis process

2.5.3 Requirement Discovery

Requirement discovery means finding all relevant information about the system. The **sources of information** for requirement gathering are: documentation, system stakeholders and specification some other system which is of similar kind. Various **methods** of requirement discovery are conducting interviews, observations, creating use case scenarios. Let us discuss **various methods** of requirements discovery.

1. Interviewing

This is an effective method of requirement gathering. The requirement engineering team communicates the stakeholders by asking them various questions about the system and its use. From the answers the requirements can be identified. There are two types of interviews. (Refer Fig. 2.5.2.)

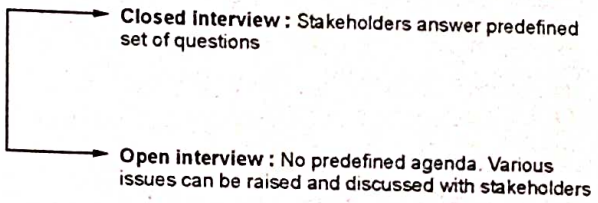


Fig. 2.5.2 Types of interview

Interviews are useful for understanding stakeholders but they are not much useful for understanding the application domain.

Characteristics of effective interviews

1. The interviews should be conducted in a free environment and they should be conducted with open minded approach. The requirement engineers should listen to stakeholders with patience. Similarly if stakeholders are expecting some unrealistic things about the system they should be ready to change their mind and ideas about the system.
2. Interviewee should start the discussion by asking questions and the requirements should be gathered together.

2. Scenario

- Scenario is the sequence of interactions made by the user with the software systems. Requirements engineers can use the information gained from this scenarios to formulate actual system requirements. The scenarios describe the interaction sessions.
- Each scenario covers one or more interaction sessions.
- Each scenario includes -
  - A description of what the system and the users want at the beginning of the scenario
  - A description of normal flow of events
  - A description of something that went wrong and handling of it
  - A description of other activities occurring in parallel
  - A description of finish state

For example : **Scenario for an article downloading in the library system.**

Initial Assumption	**User logs in to the library system and searches for the desired article**
Normal flow of events	1. When user find the required article and clicks the button for downloading the article, he might be displayed for either provide the subscriber information or request for payment for purchasing the article. 2. User either submits the subscription information or pays online for the article. 3. He might be displayed with copyright information and might be requested to fill up the copyright related form 4. User fills up the form and submits it. 5. The user is authenticated an required document can be presented to the user in pdf form. 6. User might select the print option for printing the required article
What went wrong	1. User fails to pay for the article 2. User makes mistakes in filling up the copyright form Both these events are handled by rejecting the download request.
Other activities	User can download other articles He might search for other articles
Finish State	User gets log out the library system.

3. View point

Viewpoint provides the perspective to the system and using these perspectives the requirements can be discovered. The viewpoint is also useful in classifying the stakeholders. Following are the types of viewpoints -

Interactor viewpoint : This viewpoint is useful for finding the interaction one system with other system. For example in ATM system the user of ATM is the interactor for the bank.

Indirect viewpoints : The user who is not using the system directly but its existence reflects on the requirements of the system. Such stakeholders form indirect viewpoint.

Domain viewpoints : The domain characteristics and constraints that affect on requirements of the system sets the domain viewpoints. For example in Library system the rules that are to be followed for reserving the book(the fine or dues should be paid already or book must be returned within 1 week etc.) form the domain view point.

From these viewpoints the requirements can be discovered. From interactor viewpoint the system requirements can be obtained. The indirect viewpoints help in getting the organizational requirements and constraints. The domain requirements provide domain constructs that need to be applied for the system.

View points in library management system is given by following Fig. 2.5.3

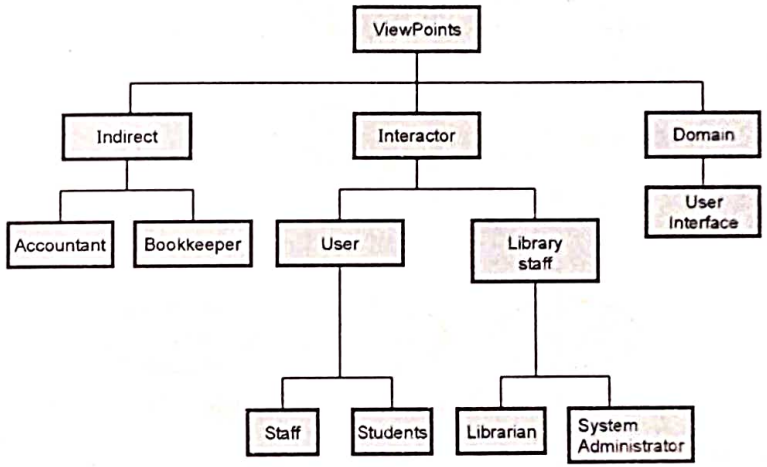


Fig. 2.5.3 Viewpoints for library system

4. Use cases

Use cases are the fundamental units of modelling language, in which functionalities are distinctly presented. The use case is a scenario based technique. Use cases help to identify individual interactions with the system. Use-cases are extensively used for requirements elicitation. By designing the proper use cases for different scenarios major reuirements of the system can be identified. The typical notations used in the use cases are as shown in Fig. 2.5.4.

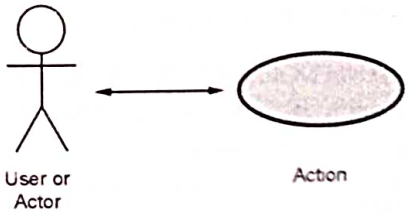


Fig. 2.5.4 Use-case notation

The use case for ATM system is as shown below.

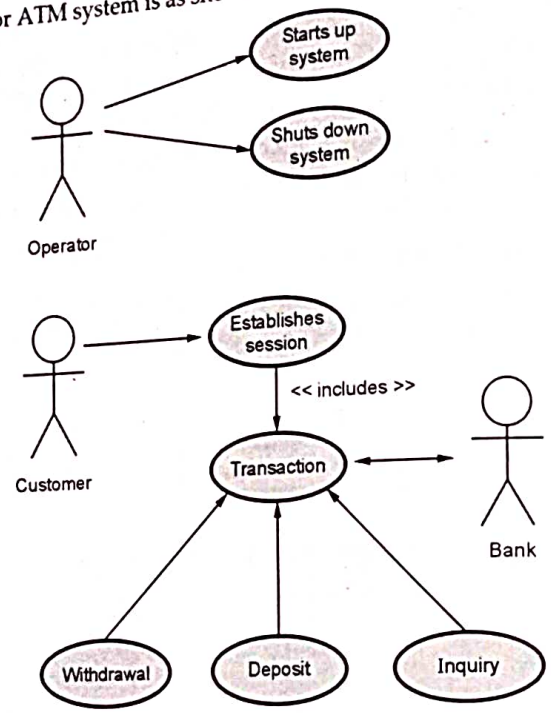


Fig. 2.5.5 Use cases for ATM system

Flow of events can be described as

System start up : The system is started when the operator turns on the switch. Initially the operator has to enter some amount of money in the cash dispenser. The connection with the bank gets established. Then only the servicing customer can begin.

System shutdown : The operator can shutdown the system only after confirming that there is no customer currently operating the ATM system. Then the connection to the bank gets disconnected.

Session : This use case starts when the user inserts the card. ATM pulls the card inside and reads it. Then further inquiry information is displayed on the display panel .

Transaction : The transaction use case is comprised of amount withdrawal, deposit and inquiry.

The interaction diagram for system startup is as given below.

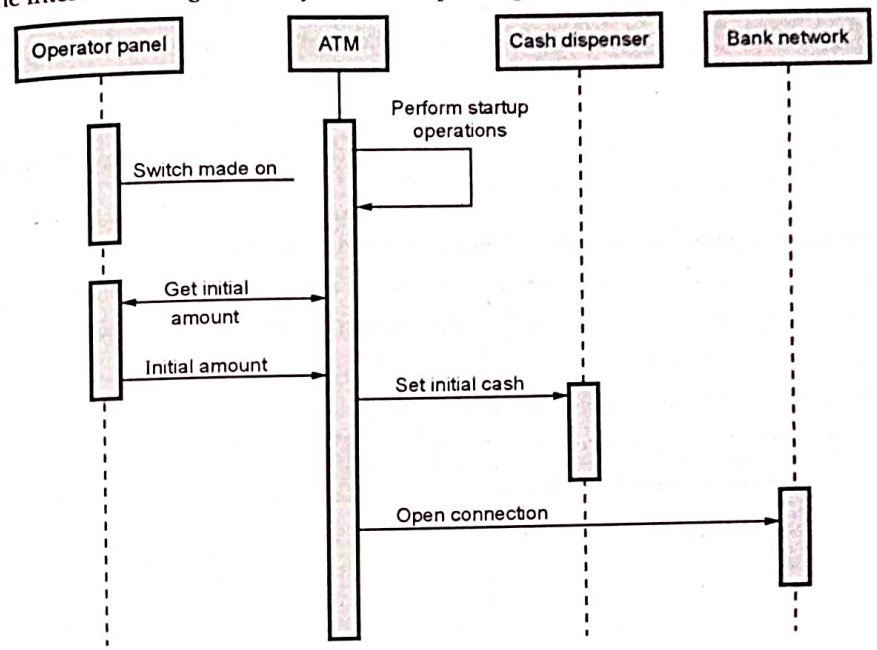


Fig. 2.5.6 Interaction diagram

This diagram depicts the sequence of operations that must be performed in order to accomplish the task.

5. Ethnography

Ethnography is a technique of observation which is used to understand social and organizational requirements.

The system analyst imagines himself as a part of the working environment in which the system will be used. He then observes the day-to-day activities and notes down the needs and tasks required by the organization. The analysts thus finds the implicit requirements of the system.

People understand their own work, but they do not understand the relationship of that work with organization. Hence by ethnographic technique such requirements can be exposed.

Two types of requirements can be discovered by ethnography.

1. Requirements obtained from working style of people

These are the requirements that can be identified from the sequence of actions that a user is performing. For example in ATM system when the user enters the PIN, the card validity action takes place. Unless and until the card gets validated there should not be any transaction processing. That means, it is required that a valid card, valid PIN must be entered for getting the money from ATM.

2. Requirements obtained from inter-activities performed by the people

Sometimes for finding the social requirements the other people's activities should be known. For example in ATM system the operator can not shutdown the system if some transaction is in processing.

Enthography can be combined with prototyping

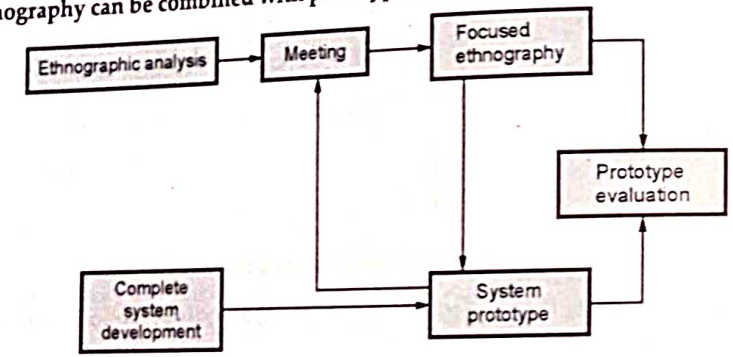


Fig. 2.5.7 Ethnographic process

The ethnography informs the development of protoype using this information prototype refinement cycles can be used.

Prototype focuses on ethnography by identifying problems and questions discussed by ethnographer.

Requirement elicitation problem

1. Stakeholders don't know what they want.
2. Stakeholders may have unrealistic expectations.
3. Stakeholders use their own language.
4. Different stakeholders may have different requirements.
5. Certain political factors may affect the requirements.
6. Business or economic changes create dynamic environment.

Example 2.5.1 *Perform requirements elicitation for watch system that facilitates to set time and alarm.*

Solution : Requirements elicitation is done for watch system by using following steps

1) **Requirements discovery :** By effective communication with customers the requirements can be identified.
2) **Requirements classification :** All unstructured requirements can be classified systematically and can be arranged in functional and non-functional requirements.
3) **Requirements prioritisation :** Some requirements are conflicting requirements. Hence they need to be priortised first. If some requirements are un-realistic, then they must be eliminated and only realistic requirements are collected.

The functional and nonfunctional requirements for watch system are -

Functional requirements

1) System allows to set time in 12:00 or 24:00 hour display format.
2) It allows to set alarm.
3) System allows to delete the already set alarm.
4) System allows to set timings for snooze.
5) System allows to stop alarm, during alarming.
6) The volume of alarm can be set by user.
7) It should allow to set more than one alarms.

Non-functional requirements

1) It should indicate battery low message if battery is low.
2) The system must be reliable and nobody should hack the system.

Review Questions

1. *What is requirements elicitation ? Briefly describe the various activities performed in requirements elicitation phase with an example of a watch system that facilitates to set time and alarm.* **AU : Dec.-16, Marks 16**
2. *What is requirement engineering ? State its process and explain requirements elicitation problem.* **AU : May-08, Marks 8**
3. *What is requirement elicitation ? Briefly describe various activities performed in requirements elicitation phase with an example of watch system that facilitates to set time and alarm.* **AU : Dec.-16, May-18, Marks 16**
4. *Write a note on what are difficulties in elicitation, requirement elicitation.* **AU : May-17, Marks 5**

5. *List any two techniques used for eliciting requirements. Compare the two techniques and list where each is applicable.* **AU : May-19, Marks 13**
6. *Write in detail about requirements gathering techniques.* **AU : Dec.-22, Marks 7**

2.6 Software Requirement Specification

AU : Dec.-11, 19, May-13,14,16,19,22, Marks 16

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is not a design document. As far as possible, it should set of what the system should do rather than how it should do it.

Software Requirements Specification

The software requirements provide a basis for creating the Software Requirements Specifications (SRS).

The SRS is useful in estimating cost, planning team activities, performing tasks, and tracking the team's progress throughout the development activity.

Typically software designers use IEEE STD 830-1998 as the basis for the entire Software Specifications. The standard template for writing SRS is as given below.

Document Title

Author(s)
Affiliation
Address
Date
Document Version

1. Introduction

1.1 Purpose of this document

Describes the purpose of the document.

1.2 Scope of this document

Describes the scope of this requirements definition effort. This section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs.

1.3 Overview

Provides a brief overview of the product defined as a result of the requirements elicitation process.

2. General description

- Describes the general functionality of the product such as similar system information, user characteristics, user objective, general constraints placed on design team.
- Describes the features of the user community, including their expected expertise with software systems and the application domain.

3. Functional requirements

This section lists the functional requirements in ranked order. A functional requirement describes the possible effects of a software system, in other words, *what* the system must accomplish. Each functional requirement should be specified in following manner

- **Short, imperative sentence stating highest ranked functional requirement.**

1. **Description**
   A full description of the requirement.
2. **Criticality**
   Describes how essential this requirement is to the overall system.
3. **Technical issues**
   Describes any design or implementation issues involved in satisfying this requirement.
4. **Cost and schedule**
   Describes the relative or absolute costs of the system.
5. **Risks**
   Describes the circumstances under which this requirement might not able to be satisfied.
6. **Dependencies with other requirements**
   Describes interactions with other requirements.
7. **... any other appropriate**

4. Interface requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

- **4.1 User Interfaces**
  Describes how this product interfaces with the user.

- **4.1.1 GUI**

  Describes the graphical user interface if present. This section should include a set of screen dumps to illustrate user interface features.

- **4.1.2 CLI**

  Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

- **4.1.3 API**

  Describes the application programming interface, if present.

- **4.2 Hardware Interfaces**

  Describes interfaces to hardware devices.

- **4.3 Communications Interfaces**

  Describes network interfaces.

- **4.4 Software Interfaces**

  Describes any remaining software interfaces not included above.

5. Performance requirements

Specifies speed and memory requirements.

6. Design constraints

Specifies any constraints for the design team such as software or hardware limitations.

7. Other non-functional attributes

Specifies any other particular non functional attributes required by the system. Such as :

7.1 Security

7.2 Binary Compatibility

7.3 Reliability

7.4 Maintainability

7.5 Portability

7.6 Extensibility

7.7 Reusability

7.8 Application Compatibility

7.9 Resource Utilization

7.10 Serviceability

... others as appropriate

8. Operational scenarios

This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations.

9. Preliminary schedule

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates.

10. Preliminary budget

This section provides an initial budget for the project.

11. Appendices

11.1 Definitions, Acronyms, Abbreviations

Provides definitions terms, and acronyms, can be provided.

11.2 References

Provides complete citations to all documents and meetings referenced.

2.6.1 Characteristics of SRS

Various characteristics of SRS are

- **Correct** - The SRS should be made up to date when appropriate requirements are identified.
- **Unambiguous** - When the requirements are correctly understood then only it is possible to write an unambiguous SRS.
- **Complete** - To make the SRS complete, it should be specified what a software designer wants to create a software.
- **Consistent** - It should be consistent with reference to the functionalities identified.
- **Specific** - The requirements should be mentioned specifically.
- **Traceable** - What is the need for mentioned requirement? This should be correctly identified.

2.6.2 Example of SRS

Software Requirements Specification For Attendance Maintenance System

Prepared by Anjali
December 1, 2014
Release 1.0

Version 1.0

Table of Contents

1. Introduction 1
   1.1 Purpose 1
   1.2 Scope 1
   1.3 Overview 1
2. General Description 1
   2.1 User Manual 1
3. Functional Requirements 1
   3.1 Description 2
   3.2 Technical Issues 2
4. Interface Requirements 2
   4.1 GUI 3
   4.2 Hardware Interface 3
   4.3 Software Interface 3
5. Performance Requirements 4
6. Design Constraints 4
7. Other Non-functional Attributes 4
   7.1 Security 5
   7.2 Reliability 5
   7.3 Availability 5
   7.4 Maintenability 5
   7.5 Reusability 5
8. Operational Scenarios 5
9. Preliminary Schedule 6

1. Introduction

1.1 Purpose

This document gives detailed functional and non functional requirements for attendance maintenance system. The purpose of this document is that the requirements mentioned in it should be utilized by software developer to implement the system.

1.2 Scope

This system allows the Teacher to maintain attendance record of the classes to which it is teaching. With the help of this system Teacher should be in a position to send e-mail to the students who remain absent for the class. The system provides a cumulative report at every month end for the corresponding class.

1.3 Overview

This system provides an easy solution to the Teacher to keep track of student attendance, and statistics.

2. General description

This attendance maintenance system replaces the traditional, manual attendance system by which a lot of paper work will be reduced. The Teacher should able to view photograph of a student along with his attendance in his Laptop. This is the primary feature of this system. Another feature is that Teacher can be allowed to edit particular record at desired time. The system should produce monthly attendance report. And there should be facility to send an e-mail/warning to the student remaining absent in the class.

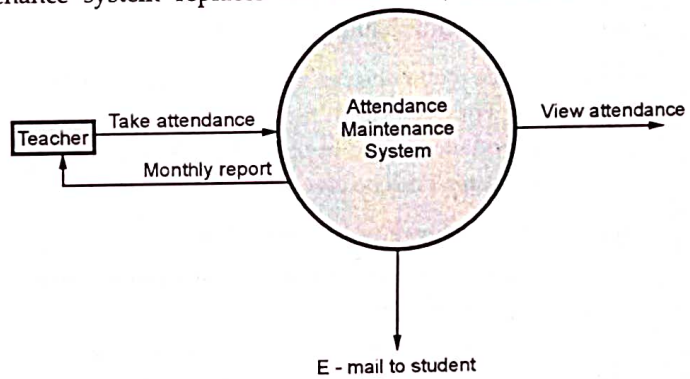


Fig. 2.6.1

Every Teacher should have Laptop with wireless internet connection. A Teacher may teach to different classes and a separate record for the corresponding classes should be maintained.

2.1 User Manual

The system should provide **Help** option in which how to operate the system should be explained. Also hard copy of this document should be given to the user in a booklet form.

3. Functional Requirements

3.1 Description

The identity of student is verified and then marked present at particular date and Time. The system should display student photograph along with their names for that corresponding class. The student may be marked present or absent depending upon his presence. The system should send e-mails to absent students. A statistical report should display individual's report or a cumulative report whenever required.

3.2 Technical Issues

The system should be implemented in VC++

4. Interface Requirements

4.1 GUI

GUI 1 : Main menu should provide options such as File, Edit, Report, Help.

GUI 2 : In File menu one can create a new record file or can open an existing record file. For example : If file *SECOMP_SEMI_07* is opened then we may view attendance record of SE COMPUTER class in year 2007 and a record for semester-I can be viewed.

GUI 3 : The display of record should be

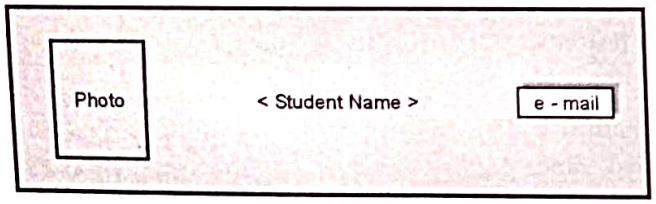


Fig. 2.6.2

The photo can be clicked to mark student present for particular class. The **e-mail** button can be clicked to send e-mail to the student being absent.

GUI 4 : Report option should display statistical report. It may be for particular student or for the whole class.

GUI 5 : Help option should describe functionality of the system. It should be written in simple HTML.

4.2 Hardware Interface

Hardware Interface 1 : The system should be embedded in the laptops.

Hardware Interface 2 : The laptop should use wireless Ethernet card to send e-mails. These Laptops should be the clients of departmental database server.

4.3 Software Interface

Software Interface 1 : Attendance maintenance system.

Software Interface 2 : The attendance database should be transmitted to departmental database server.

Software Interface 3 : E-mail message generator which generates standard message of absence.

Software Interface 4 : Report generators

5. Performance requirements

This system should work concurrently on multiple processors between the college hours. The system should support 50 users.

The e-mail should be sent within one hour after the class gets over.

The system should support taking attendance of maximum 100 students per class.

6. Design constraints

The system should be designed within 6 months.

7. Other non-functional attributes

7.1 Security

The teacher should provide password to log on to the system. He/she should be able to see the record of his/her class.

The password can be changed by the Teacher by providing existing password.

7.2 Reliability

Due to wireless connectivity , reliability can not be guaranteed.

7.3 Availability

The system should be available during college hours.

7.4 Maintenability

There should be a facility to add or delete or update Teachers and students for each semester.

7.5 Reusability

The same system will be used in each new semester.

8. Operational scenarios

There will be student database, Teacher database. The student database will contain students name, class, attendance, e-mail address, address, phone number.

The Teacher database will contain Teacher's name, classes taught, e-mail address, phone number.

9. Preliminary schedule

The system has to be implemented within 6 months.

Importance of SRS : The software requirements provide a basis for creating the Software Requirements Specifications (SRS).

The SRS is useful in estimating cost, planning team activities, performing tasks, and tracking the team's progress throughout the development activity.

Example 2.6.1 *An independent trunk company wants to track and record its drivers driving habits. For this purpose the company has rented 800 phones numbers and has printed the numbers in the front, back and sides of all trucks owned by the company. Next to the 800 numbers a message is written "PLEASE REPORT ANY DRIVER OR TRUCK PROBLEM BY CALLING THIS NUMBER". The hacking company waits for you to develop a system that :*

i) Collects information from caller about the driver performance and behaviour as well as truck condition,

ii) Generates daily and monthly reports for each driver and truck management.

iii) Reports problems that require immediate action to an on-duty manager.

Analyse the problems that statement and list major functions to be incorporated with the SRS document. **AU : Dec.-11, Marks 8**

Solution : The major functions to be included with SRS document are -

i) A) Creation of databases such as drivers database, trucks database, callers database.

B) Updation for above created databases.

C) Display and searching operations on database.

ii) A) Report generation (Daily) of trucks and drivers.

B) Report generation (Monthly) of trucks and drivers.

iii) A) Sending of SMS to corresponding driver about the problem.

B) Updation for duty hours if necessary.

C) Deletion of particular record (For truck database) if the truck is out of condition.

D) Insertion of new record (New truck arrival).

Example 2.6.2 *Develop the software requirements document for the following requirement. A coffee vending machine serves coffee to customers. A customer can choose a type of coffee among a list often options, supply the amount required and get served. Each coffee is prepared by adding units of hot water, coffee powder, milk and sugar. The recipe for each coffee is stored.* **AU : May-19, Marks 13**

Solution :

1. Introduction

1.1 Purpose

This document describes the interaction between the user and Coffee Vending Machine(CVM). The purpose of this document is that the requirements mentioned in it should be utilized by software developer to implement the system.

1.2 Scope

This system allows the customer to interact the vending machine. The purpose of the CVM is to accept payment for the customer, dispense coffee (this will have the options of added milk and / or sugar) and return any excess money to the customer as change.

1.3 Overview

This system dispenses the coffee of a user's choice.

2. General description

Coffee vending machine is a type of vending machine that dispenses hot coffee. Some of the machines, particularly older models, utilize powdered instant coffee mixed with hot water, and some of these offer condiments such as cream and sugar. Some models fresh-brew the coffee using hot water and ground coffee beans, and some also grind the coffee to order using coffee grinders installed in the machines. Some modern machines also provide other hot drinks such as tea, espresso, lattes, cappuccinos, mochas and hot chocolate. Some of the machines dispense canned coffee, and some dispense both hot coffee and iced coffee.

These coffee vending machines typically require payment, functioning as coin-operated machines.

2.1 Application Document

The document describes the interaction between the customer and the CVM. This document also provides the information about maintenance of the system.

3. Functional requirements

3.1 Description

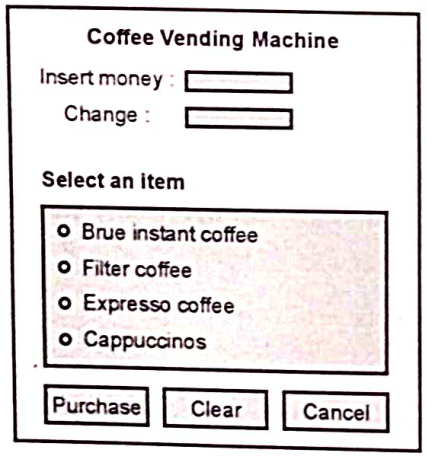
A coffee vending machine serves coffee to customers. A customer can choose a type of coffee among a list often options, supply the amount required and get served. Each coffee is prepared by adding units of hot water, coffee powder, milk and sugar. The recipe for each coffee is stored.

3.2 Technical Issues

The system should be implemented in Java

4. Interface requirements

4.1 GUI



4.2 Hardware Interface

Hardware Interface 1 : The system should be embedded in the laptops.

4.3 Software Interface

Software Interface 1 :

Interface Identification: Customer

Interface type: Person

Description : The customer is an actor who uses the Vending machine system to do the transactions

Software Interface 2 :

Interface Identification : Maintainer

Interface type : Person

Description : The maintainer is an actor who maintains and manage the vending machine system.

Software Interface 3 :

Interface Identification : Controller

Interface type : System

Description : The admin is an actor who controls the vending machine.

5. Performance requirements

Not Applicable,

6. Design constraints

The system should be designed within 6 months.

7. Other non-functional attributes

7.1 Security

Not applicable.

7.2 Reliability

If there exists a situation in which insufficient payment has been tendered, to meet the price of the chosen item. It is handled by refusing to dispense the required beverage until additional payment is inserted, sufficient to equal or exceed the cost of the selected beverage.

7.3 Availability

The coffee must be made available on correct payment.

7.4 Maintenability

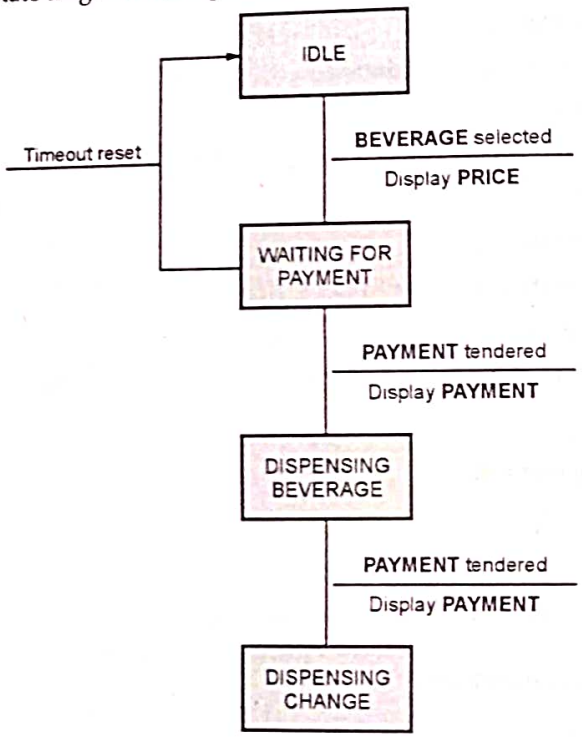
There should be a facility to clear the choice that is been made.

7.5 Reusability

The same system will be used for each type of coffee making.

8. Operational scenarios

Following is a state diagram that represents the operational scenario of the system.



Event List

The event list consists of three types of event : Direct flow of events, indirect flow of events, temporal(driven by clock or schedule)event.

Direct Event

(1) Customer makes the choice for the coffee.

(2) Customer tenders the payment.

(3) Coffee machine dispenses coffee.

(4) Coffee machine dispenses change.

Indirect Event

(1) Payment exceeds price

(2) Beverage not available.

(3) Hot water not available,

(4) Drinkable water not available.

Temporal Event

(1) Maintenance inspects and resupplies coffee machine

9. Preliminary schedule

The system has to be implemented within 6 months.

Example 2.6.3 *Prepare a software requirement specification document for "Library Management System"*

AU : Dec.-19, Marks 15

Solution :

1. Introduction

1.1 Purpose

This document gives the detail functional and non functional requirements for Library management system. Thee purpose of this document is that the requirements mentioned in it should be utilized by the software developer to implement the system.

This SRS will be used by the software engineers who are constructing the system and the users who will use the system. The hotel end users will be able to use this SRS as a test to see if the software engineers will be constructing the system to their expectations.

1.2 Scope

The library management system will automate the major library operations. Various subsystems will be student accounts management system, Book issuing system, Stock maintenance systems and fine calculation and management system.

1.3 Definitions, Acronyms and Abbreviations

- LMS - Library Management System
- SRS - Software Requirement Specification
- GUI - Graphical User Interface
- Stakeholder - The person who will participate in this system. For example Student, Staff, Librarian.

1.4 Overview

This system provides an easy solution for the students, staff and Librarian for accessing and managing the books in the library.

2. Overall description

The Library management system is developed for handling the activities for various users such as Student, staff, librarian and library staff.

2.1 Product Perspective

This is a standalone system.

2.2 Hardware Interfaces

This system is placed in the Personal Computer in the library.

2.3 Software Interfaces

There are various databases in the system. These databases can be created in Oracle or MySQL. The hotel books and Student/Staff information is maintained by the LMS.

The book database include the Title of the book, ISBN number, publisher, edition, number of copies, price,status(Old or new). The Student/staff database will include information such as first name, last name, address, phone number, designation(for staff) or class(for student), name of books issued, Library Card number, expects date of book return, fine to be paid(if any).

3. Functional requirements

The functional requirement will define the fundamental functioning that the system should perform.

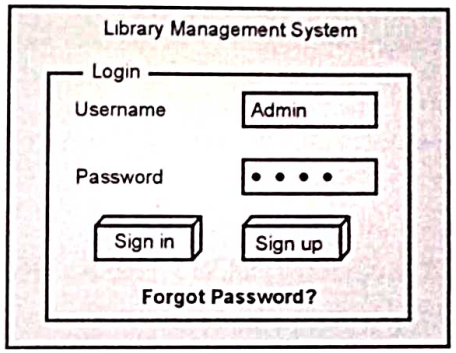
1. The system should record all the details of the student, staff and book.
2. The system should display the search results if the user searches for some book or article.
3. The system should allow to select the book for issuing.
4. The system should record the expected issue date and time of book.
5. The system should allow the modification in the student/staff/book information without reentering the entire information.
6. The system should allows to create an account for the new student.
7. The system should allow to cancel the account if the student leaves the college.
8. The system should display the availability of book or magazine.
9. The system shall allow to assign password to the users.
10. The system should keep track of the purchased items stock.
11. The member is allowed to reserve a book if it is currently unavailable.
12. The system should generate daily or monthly report for the stock.

13. The system should allow addition of book/magazine records if new books are purchased.
14. The system should allow deletion of book/magazine records if the books are old and in poor condition.

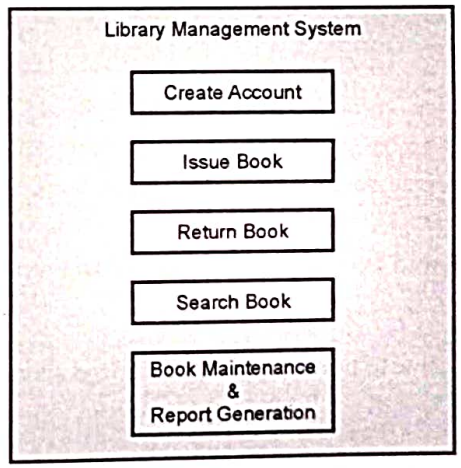
4. Interface requirements

4.1 GUI

GUI 1 : The login screen will be displayed so that the stakeholder of the system will enter the user name and password.



GUI 2 : The main menu will be displayed for Create Account, Search, Issue/Return Book, Book maintenance and Report generation.



GUI 3 : When the Create Account button is clicked then the form will be displayed. This form will allow the user to enter the Student/Staff details such as last name, first name, phone number, class and so on.

GUI 4 : If the user selects for the Search then the user will be allowed to search for the desired book.

4.2 Hardware Interfaces

The system should be embedded in the Computers in the library.

4.3 Software Interface

The system must be interfaced with Oracle or Access database.

5. Performance requirements

The performance requirement defines the response time for system functionality.

The load time for the GUI should not be more than four seconds.

The log in must be verified within 7 seconds.

The search query should be processed with in 3 seconds and response must be given.

6. Design constraints

The system must be designed as a standalone system and must be run on window based system. The system can be developed in Java or Visual Basic. The database can be implemented in Oracle or MySQL.

7. Other non functional attributes

7.1 Security

The System administrator/Librarian must provide password to log on the system. The password can be changed by the system administrator/Librarian. The Student should be allowed to the access the book database for searching of book. He/She is not allowed to modify the database.

The student/Staff is allowed to change their passwords for security purpose.

7.2 Reliability

The system must be reliable to prevent any unauthorized access.

7.3 Availability

The system should be available during the College hours.

7.4 Maintainability

There should be the facility to add/modify information about the Books, Student and Staff.

8. Operational scenarios

If a new students arrives in the library for membership then the librarian creates an account; issues him the library Card, User name and password.

If the student already has an account, he searches for the desired book/magazine. If the book is available then he makes the request to the librarian for issuing the book. The librarian issues the book and updates the database.

If the book is not available then the member reserves the book. On availability of the book, it is issued to the customer.

If the student/staff does not return the book on the specified date then he has to pay the fine.

The librarian updates the stock and the system generates the monthly report. The old and outdated books are removed from the library and accordingly the stock must be updated.

If new purchase is made then also librarian is allowed to update the book database.

9. Preliminary schedule

The system must be implemented within 6 months.

Review Questions

1. *Show the template of IEEE standard software requirements document.* **AU : May-13, Marks 8**
2. *What are the components of the standard structure for the software requirements document ? Explain in detail.* **AU May-14, Marks 8, May-18, Marks 16**
3. *Write the software requirement specification for a system of your choice.* **AU : May-14, Marks 8**
4. *Explain the organization of SRS and highlight the importance of each subsection.* **AU : May-16, Marks 8**
5. *Describe how Software requirements are documented. State the importance of documentation.* **AU : Dec.-20, Marks 13**
6. *What are the components of software requirement specification document? Explain.* **AU : May-22, Marks 7**

2.7 Formal System Specification

- A formal system specification for software requirements is an **unambiguous software system description.** In the formal system specification, **formal methods** are used to represent the requirements of the system.
- Formal methods provide us with tools to precisely describe a system and show that a system is correctly implemented. We say a system is correctly implemented when it satisfies its given specification.
- The specification of a system can be given either as a list of its **desirable properties** (property-oriented approach) or as an **abstract model** of the system (model-oriented approach).

2.7.1 Concept of Formal Technique

- A formal technique is a mathematical method used to specify a hardware or software system.
- The formal methods verify that the implementation satisfies the requirements specification, they prove the properties of the system without necessarily running the system.
- The formal specification language consists of the **syntactic domain, semantic domain** and a relationship called **satisfaction relation.**
- The formal techniques can be used at every stage of the software development life cycle (requirement specification, design, coding and implementation) to verify that one stage's output conforms to the previous stage's output.

Semantic domain

- Abstract Data Type(ADT) specification languages are used to specify algebras and programs.
- The programming languages are used to specify functions from input to output values.
- The distributed system specification languages are used to specify state sequences, event sequences, state transition sequences and finite state machines.

Syntactic domain

- The syntactic domain of formal specification language consists of alphabets, symbols and set of rules.
- These rules are used to construct **well-formed formula** using alphabets. Basically, these well formed formulas are used to specify the system.

Satisfaction relation

- For any model of a system, it is important to determine if elements of the semantic domain satisfy its specification. This satisfaction is determined by a function which is known as **semantic abstraction function.** The semantic abstraction function maps the elements of semantic domain to equivalent classes
- There are different specifications that are used to describe different aspects of the system. For example - a specification for describing system behavior or a specification describing the system structure.

2.7.2 Merits and Limitations of Formal Methods

Merits

1) Formal methods provide a **precise and unambiguous** way to describe the behavior and properties of a software system.
2) Formal methods can be used to **prove the correctness of a software system** concerning its specification. This is particularly valuable for safety-critical and mission-critical systems.
3) Formal specification can serve as a **clear and comprehensive documentation** of system requirements.
4) The formal methods can **identify errors in the specification** itself. This allows corrections before implementation.
5) The formal methods **allow rigorous analysis** of complex systems. Hence formal methods promote the construction of rigorous specification of the system.
6) The mathematical basis of formal methods facilitates **automating the analysis of specifications.** The possibility of automatic verification is one of the most important advantages of formal methods.

Limitations

1) Formal methods are **difficult to learn and use.**
2) It is **difficult to check the absolute correctness** of systems using theorem-proving techniques.
3) Formal techniques are **not able to handle complex problems.**
4) It can be challenging to create complete formal specifications that capture all aspects of a **real-world system.**

2.7.3 Model-Oriented Approach and Property-Oriented Approach

- The formal method is classified using two approaches - **Model oriented approach** and **Property-oriented approach**.

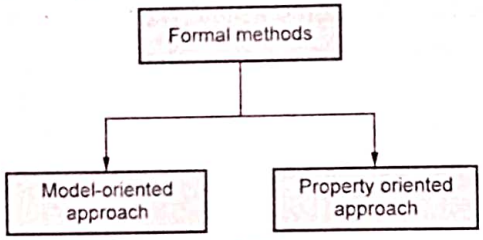


Fig. 2.7.1

- **In model oriented approach,** the system's behavior is represented by directly constructing a model of the system with the help of mathematical structures such as tuples, relations, functions, sets and sequences.
- **In property property-oriented approach,** system's behaviour is defined indirectly by stating its properties. These properties are specified in terms of a **set of axioms**.
- The property-oriented approach is more suitable for **requirements specification** and model-oriented approach is more suitable for system **design specification**.

Difference between Model-oriented approach and Property oriented approach

Sr. No.	Model-oriented approach	Property oriented approach
1.	The system's behavior is represented by an **abstract model** of the system.	The system's behavior is represented by stating the **properties** or requirements of the system.
2.	This approach is more suitable for the **design specification** of the system.	This approach is more suitable for **requirements specification** of the system.
3.	Mathematical **structures** such as tuples, relations, functions, sets and sequences are used in this approach.	The **set of axioms** is used in this approach.
4.	**A limited number of implementations** are possible using the model oriented approach.	**A large number of implementations** are possible using property property-oriented approach.
5.	**Making minor changes** in the specification is very **difficult** in model-oriented approach.	**Making changes** at any specification stage is easier in a property-oriented approach.

- There property oriented approach is classified into two categories - Axiomatic specification and algebraic specification

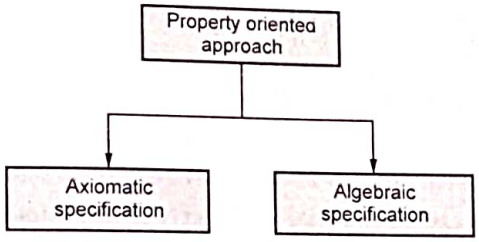


Fig. 2.7.2

2.7.4 Axiomatic Specification

- The axiomatic specification is based on formal logic or predicate calculus.
- The main purpose of this type of semantic method is - **formal program verification**
- **Axioms or inference rules** are defined for each statement type in the language to allow transformations of logic expressions into more formal logic expressions.
- An Axiom is a logical statement that is assumed to be true.
- An Inference Rule is a method of inferring the truth of one assertion based on the values of other assertions.

$$\frac{S_1, S_2, S_3, \ldots, S_n}{S}$$

- The **rule** states that if S_1, S_2, ... and S_n are true, then the truth of S can be inferred. The top part of an inference rule is called its **antecedent**; the bottom part is called it **consequent**.

Concept of assertion :

- The logic expressions are called **assertions**.
- An assertion before the statement or command is called a precondition. This condition states the relationships and constraints among variables that are true at that point in execution.
- An assertion following a statement is called postcondition.

<precondition> statement <postcondition>

- The axiomatic semantic is specified in the following manner

{P} statement {Q}

Where P is a precondition, Q is the postcondition.

- For example -

$$x = y+1 \qquad \{x>1\}$$

One possible precondition : $\{y > 10\}$

Axiomatic specification for assignment statement

The precondition and postcondition of an assignment statement together define precisely its meaning.

Let x = E be a general assignment statement and N be its postcondition.

Then, its precondition, M, is defined by the axiom

$$M = N_x \rightarrow E$$

That means M is computed as N with all instances of x replaced by the expression E.

How to develop axiomatic specifications ?

Step 1 : Establish the range of input values over which the function should behave correctly.

Step 2 : Establish the constraints on the input parameters as a predicate.

Step 3 : Specify a predicate defining the condition which must hold on the output of the function.

Step 4 : Establish the changes made to the function's input parameters after execution of the function.

Step 5 : Combine all of the above into pre and post-conditions of the function.

2.7.5 Algebraic Specification

- In the algebraic specification technique, an object class or type is specified in terms of relationships that exist between the operations defined on that type.
- The algebraic specifications define a system as **heterogeneous algebra**. The heterogeneous algebra is a collection of different sets on which several operations are defined. Traditional algebras are **homogeneous**. A **homogeneous algebra** consists of a single set and several operations.
- The algebraic specification consists of four sections -
  1. **Type section :** In this section, the data types being used are specified.
  2. **Exception section :** In this section, exceptional conditions that may occur during the operations are defined.
  3. **Syntax section :** This section defines the signatures of the interface procedures. The collection of sets that form the input domain of an operator and the sort where the output is produced is called the signature of the operator.

4. **Equations section :** This section specifies the set of equations or rewrite rules. These rules define the meaning of the interface procedures.

The algebraic specification consists of the following **set of operators -**

1. **Basic construction operators :** These operators are used to create or modify entities of a type. For example - **create** and **append** are basic construction operators.
2. **Extra construction operators :** These are the construction operators other than the basic construction operators. For example - The **remove** is an extra construction operator.
3. **Basic inspection operators :** These operators evaluate attributes of type without modifying them. For example - eval, get are the basic inspection operators.
4. **Extra inspection operators :** These extra inspection operators are other than the basic inspection operators.

Example of algebraic specification

Following is an algebraic specification that represents the Cartesian coordinates. The operations defined on X and Y coordinates which evaluates the x and y attributes of an entity. The **IsEq** compares two entities for equality.

Types section

defines Coord

uses Integer, Boolean

Syntax section

Create (Integer, Integer) → Coord ;

X (Coord) → Integer ;

Y (Coord) → Integer ;

IsEq (Coord, Coord) → Boolean ;

Equations section

X (Create (x, y)) = x

Y (Create (x, y)) = y

IsEq (Create (x1, y1), Create (x2, y2)) = ((x1= x2) and (y1=y2))

Properties of algebraic specification

1) **Finite Termination Property :** This property of algebraic specification ensures that any sequence of operations applied to a data structure or system will terminate in a finite number of steps. In other words, it guarantees that there will be no infinite loops or non-terminating computations.
2) **Unique termination property :** The unique termination property of an algebraic specification states that any two sequences of operations involving the interface procedures of the specification will eventually terminate and produce the same result.
3) **Completeness :** The completeness property of an algebraic specification states that the specification is sufficient to define the behavior of the system for all possible inputs and outputs.

Pros and cons of algebraic specification

Pros

1) The algebraic specifications are based on mathematical structures. Hence they are unambiguous and specification.
2) Using algebraic specification, the effect of the arbitrary sequence of operations can be studied.

Cons

1) Algebraic specifications are hard to understand.
2) The algebraic specifications are difficult to integrate with programming languages.

2.8 Finite State Machines

- A finite state machine is a mathematical abstraction used to design algorithms.
- In simpler terms, a state machine will read a series of inputs. When it reads an input, it will switch to a different state. Each state specifies which state to switch to, for a given input.
- A very good example of a finite state system is the control mechanism of the elevator. This mechanism only remembers the current floor number pressed, it does not remember all the previously pressed numbers.
- The finite state system is a very good design tool for programs such as text editors and lexical analyzer (which is used in compilers). The lexical analyzer is a program

that scans your program character by character and recognizes those words as tokens.

2.8.1 Formal Definition

A Finite State Machine (FSM) is a collection of 5-tuple $(Q, \Sigma, \delta\, q_0, F)$ where,

Q is a finite **set of states**, which is non empty.

Σ is **input** alphabet, indicates input set.

q_0 is an **initial state** and q_0 is in Q i.e. $q_0 \in Q$.

F is a set of **final states.**

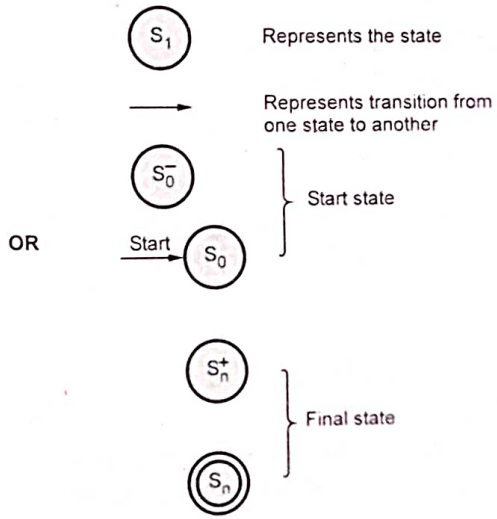
δ is a **transition function** or a mapping function. Using this function the next state can be determined.

2.8.2 Representation

A transition diagram or transition graph can be defined as collection of -

1) Finite set of states K.
2) Finite set of symbols Σ.
3) A non empty set S of K. It is called **start state.**
4) A set $F \subseteq k$ of **final states.**
5) A transition function $K \times A \rightarrow K$ with K as state and A as input from Σ^*.

The notations used in transition diagram are -



2.8.3 Types of Finite State Machines

There are two types of finite automata or finite state machine

i) Deterministic finite automata.

ii) Non deterministic finite automata.

The deterministic finite automata is deterministic in nature, in the sense, each move in this automata is uniquely determined on current input and current state. On the other hand, it is non deterministic in nature, in NFA i.e. non deterministic finite automata.

2.8.4 Applications of Finite State Machine

1. Finite automata are used in text processing in matching the pattern against the given text.
2. It is used in compilers for analyzing the syntax of the language.
3. The finite automata are also used in preparing the hardware designs.
4. The Context-Free Grammar (CFGs) are used in programming languages and artificial intelligence. Originally, CFGs were used in the study of the human languages. Cellular automata are used in the field of biology, for instance - representing the pine cones growth and pigmentation patterns.
5. Automata simulators are pedagogical tools used to teach, learn and research automata theory.
6. In switching theory, design and analysis of digital circuits automata theory is useful.

2.8.5 Deterministic Finite State Machine

The finite automata is called Deterministic Finite Automata(DFA) if there is only one path for a specific input from current state to next state. For example, the DFA can be shown as below.

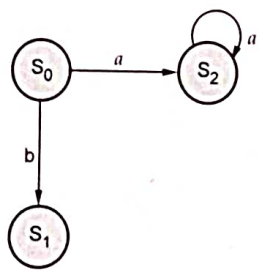


Fig. 2.8.1 Deterministic finite automata

From state S_0 for input 'a' there is only one path, going to S_2. Similarly from S_0 there is only one path for input b going to S_1.

Example 2.8.1 *Design a DFA which checks whether the given binary number is even.*

Solution : The binary number is made up of 0's and 1's. When any binary number ends with 0 it is always even and when a binary number ends with 1 it is always odd. For example,

0100 is a even number, it is equal to 4.

0011 is an odd number, it is equal to 3.

And so while designing FA we will assume one start state, one state ending in 0 and other state for ending with 1. Since we want to check whether given binary number is even or not, we will make the state for 0, the final state.

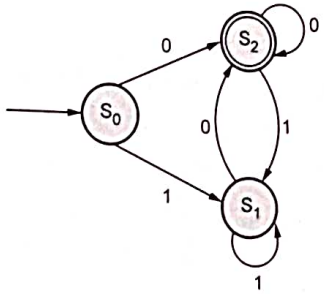


Fig. 2.8.2 For Ex. 2.8.1

The FA indicates clearly is a state which handles all the 1's and S_2 is a state which handles all the 0's. Let us take some input.

$$01000 \Rightarrow 0\underline{S_2}1000$$
$$01\underline{S_1}000$$
$$010\underline{S_2}00$$
$$0100\underline{S_2}0$$
$$01000S_2$$

Now at the end of input we are in final or in accept state so it is an even number. Similarly let us take another input.

$$1011 \Rightarrow 1\underline{S_1}011$$
$$10\underline{S_2}11$$
$$101\underline{S_1}1$$
$$1011S_1$$

Now we are at the state which is a non final state.

2.8.6 Non Deterministic Finite State Automata(NFA)

The concept of non-deterministic finite automata is exactly reverse of deterministic finite automata. The finite automata is called NFA when there exists many paths for a specific input from current state to next state. The NFA can be shown as in Fig. 2.8.3.

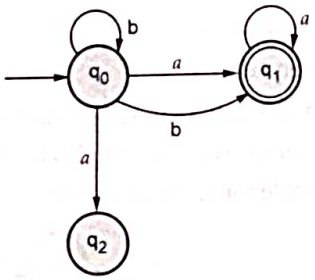


Fig. 2.8.3 Non-deterministic finite automata

Note that the NFA shows from q_0 for input a there are two next states q_1 and q_2. Similarly, from q_0 for input b the next states are q_0 and q_1.

Thus it is not fixed or determined that with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

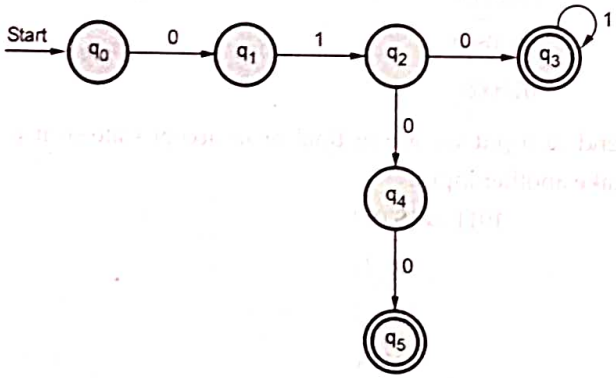
Example 2.8.2 *Construct NFA for the language*

$L = \{0101^n \cup 0100 \mid n \geq 0\}$

Over $\Sigma = \{0, 1\}$.

Solution :

Here in language L first three symbols are common i.e. 010. Hence the NFA can be drawn as



2.9 Petri Nets

AU : Dec.-19, Marks 13

- The Petri Nets are invented by Carl Adam Petri in 1962.
- The Petri Nets can be rigorously used to define the system. The Petri nets are more popularly used for distributed systems and systems with resource sharing.
- In Petri Nets there are three types of components - places (circles), transitions (rectangles) and arcs (arrows)

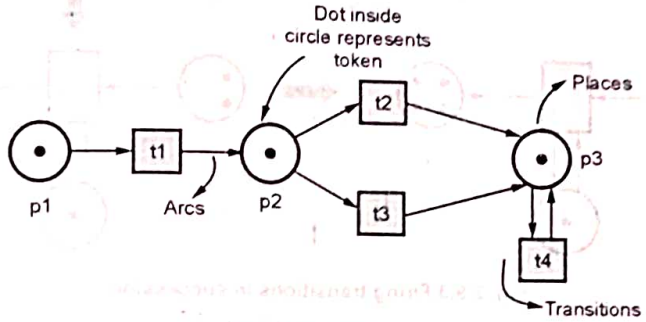


Fig. 2.9.1 Petri nets

- **Places :** Represent possible states of the system.
- **Transitions :** Cause the change in the states
- **Arc :** Connects a place with transition or transition with a place.

For example -

- Another equivalent notation used is a solid bar. This solid bar is used to denote transitions.
- The token represents the dynamic objects. The states having block dots are called marked Petri token. If there is a single dot inside the circle then that means it represent the one unit of certain resource.
- **Firing a transition :** Transitions take input from the input places and produce tokens in the output places. This is called firing a transition. For example

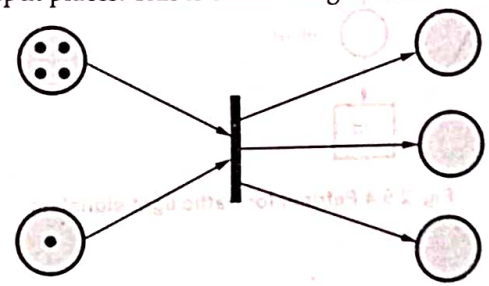


Fig. 2.9.2 Firing transitions

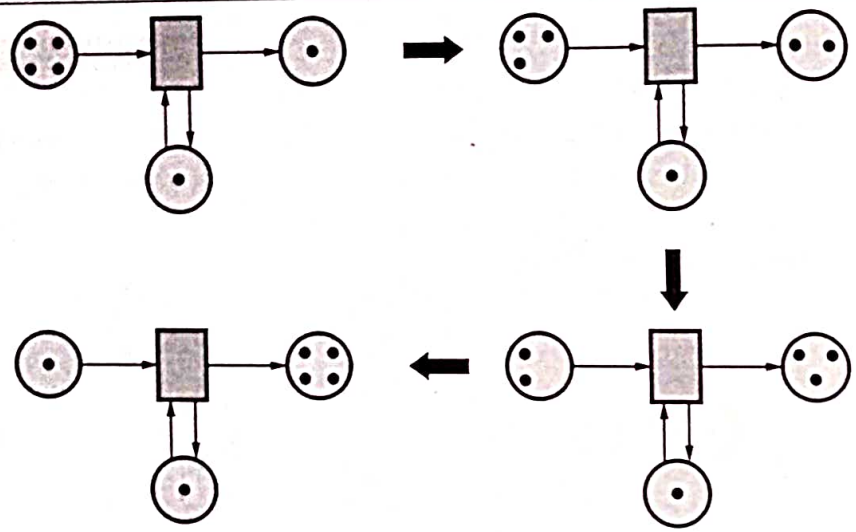


Fig. 2.9.3 Firing transitions in succession

Example 2.9.1 *Draw a petrinet for a traffic light switching system.*

Solution :

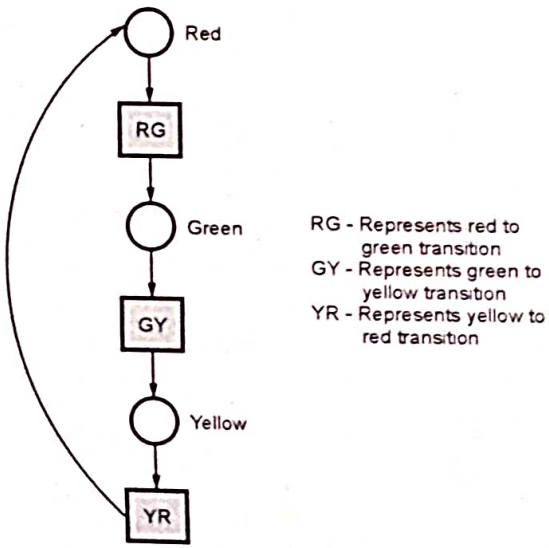
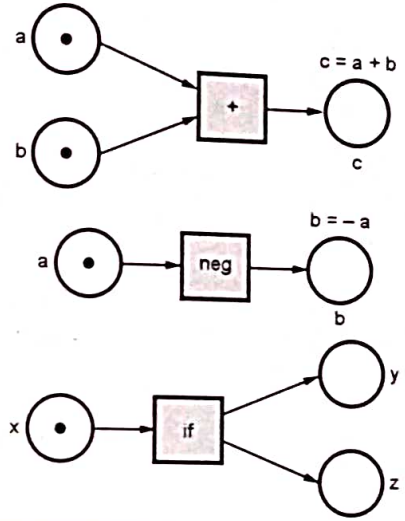


Fig. 2.9.4 Petrinet for traffic light signaling

Example 2.9.2 *Represent the operations i) $c = a + b$ ii) $b = -a$ iii) $x > 0$ then $y = x$ else $z = x$*

Solution :



Example 2.9.3 *Draw a Petri Net that depicts the operation of an "Automated Teller Machine". State the functional requirements you are considering.* **AU : Dec.-19, Marks 13**

Solution : Petri Net for demonstrating the operation of Automated Teller Machine.

(Refer Fig. 2.9.5 on next page)

Functional Requirements

1) The card should be valid
2) Pin entered by the user must be valid.
3) The amount to be withdrawn must be less than the required balance amount.
4) The balance amount must be updated and then displayed to the user after withdrawal of money.

Advantages of using Petri Nets

1. It specifies the precise definition of problem.
2. It represents the graphical workflow.
3. It is useful in allocating resources to the tasks.

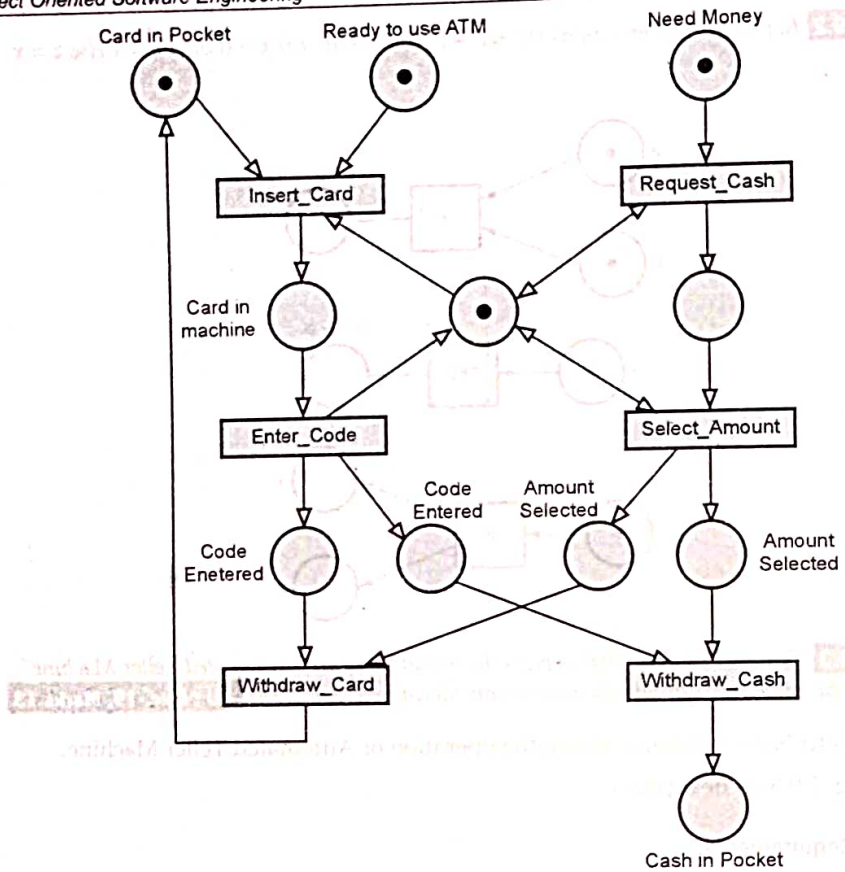


Fig. 2.9.5

2.10 Object Modeling using UML

AU : May-11,14, Dec.-18, Marks 16

Object oriented analysis is the process of identifying and describing the objects for given problem instance. For example : For designing the banking systems **customer, Account, Branch** are the objects that must be described.

Object oriented design is the process of defining the objects and their collaboration with other objects in order to fulfill the requirements. For example in **banking system getCustomerInfo, Withdraw_Amount, Deposit_Amount** and so on are the methods that help in collaborating one object with another object. These methods actually help in fulfilling requirements. The methods of objects represent the behavior of the objects. This can be illustrated by the Fig. 2.10.1.

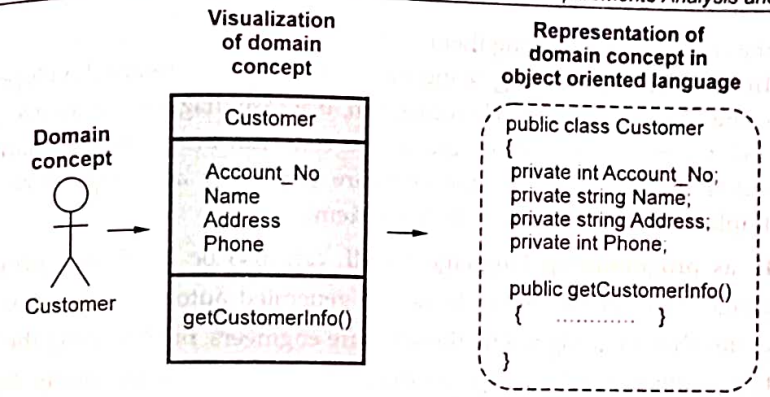


Fig. 2.10.1 Representation of object orientation

Key Point : *Object oriented analysis and design is the study of objects.*

2.10.1 What is UML?

Definition : The Unified Modeling Language (UML) is a standard diagramming notation for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

- **The goals of UML are -**
  1. UML is intended to provide ready- to use and expressive visual modeling language in order to develop the effective system model.
  2. The modeling must be independent of the programming language or platform but at the same time the automatic code that can be generated by this tool must support the modern programming languages.
  3. This tool supports the high level concepts such as collaborations, framework, patterns and components.
  4. It helps to integrate the organizational standards in the software systems.
- **According to Martin Fowler there are three ways of using UML -**
  1. **UML as sketch :** For exploring the difficult part of the system or for representing the solution to particular problem, software designer create informal diagrams.
  2. **UML as blueprint :** The UML can be used as a blueprint for two main reasons - either for reverse engineering or for forward engineering.
     - **In Reverse Engineering** the UML tool reads the source code and generates the class diagram, sequence diagram and so on. With the help of these diagrams the user can understand the structure of the system elements and

the collaborations among them.

- **In Forward Engineering**, using the UML tool the software developers draw some diagrams such as class diagram, use case diagram, sequence diagram and so on. From these diagrams the code can be generated manually or automatically. Later on, the software developer adds up more code to implement the complete software system.

3. **UML as programming language :** UML can also be used as a programming language. That means, the code can be generated automatically from the UML diagrams that are designed by the software engineers. But for using the UML tool in this manner it is essential to draw the diagrams with highly logical and practical approach. Although one can use UML as a tool for programming language it is still immature to use it in this manner.

- **UML can be applied as three perspectives -**

1. **Conceptual perspective :** Using this perspective, the things in a real world situation are described by the UML diagram.

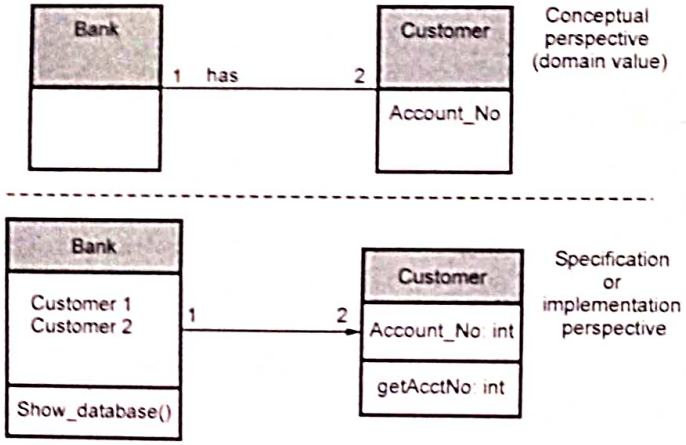


Fig. 2.10.2 Three perspectives of UML

2. **Specification perspective :** Using this perspective, the UML diagrams describe the software abstractions or components with specification or interface.
3. **Implementation perspective :** Using this perspective, the UML diagrams describe the particulars of implementation.

The most commonly used perspective is specification perspective. However, some of the software systems make use of implementation perspective.

- **Meaning of class in different perspective**

The classes used in the above described perspectives have three different names. Hence we define -

1. **Conceptual class :** This type of class represents the real world entities.
2. **Software class :** This type of class represents the specification or implementation perspective of a software component.
3. **Implementation class :** This type of class is implemented using some suitable object oriented programming language. The examples of object oriented programming languages are C++, Java.

2.10.2 UML Diagrams

- Diagrams are the **graphical representation** of the set of the elements.
- The diagrams can be drawn to represent the system from **different perspectives.**
- The same elements may appear in different diagrams.
- The diagram contains the collection of elements and relationship among these elements.
- Various **types of diagrams** that can be drawn in UML are -

1. Class diagram	2. Object diagram	3. Component diagram
4. Use case diagram	5. Sequence diagram	6. State diagram
7. Activity diagram	8. Deployment diagram	9. Package diagram

- **Class diagram :** It contains the set of **classes, interfaces, collaborations** and their **relationships**. The class diagrazm represents the **static design view** of the system. These diagrams are commonly drawn in UML.
- **Object diagram :** These diagrams show the **relationship among the different objects**. Objects are the instances of the classes, hence the object diagrams represent real or **prototypical cases**. These diagrams represent the **static design view** of the system.
- **Component diagram :** This diagram contains the collection of **components** and **connectors**. It represents the **static design view** of the system. It shows the encapsulated classes, interfaces, ports and internal structure of the system.
- **Use case diagram :** These diagrams are drawn to represent the **static design view** of the system. The use cases represent the **behaviour** of the system. It consists of collection of **use cases, actors**(special type of classes) and their **relationships**.

- **Sequence diagram :** These are kind of interaction diagrams consisting of set of **objects, roles** along the **messages**. They are specially useful for modeling the behaviour of the system.
- **State diagram :** The state diagrams represent the **states, transitions, activities** and **events**. The event-ordered behaviour of the object is emphasized.
- **Activity diagram :** This diagram represents the **flow of control** or data step by step occurring in computations. These diagrams emphasize on representation of functionality. These diagrams represent the **dynamic view** of the system.
- **Deployment diagram :** It addresses the static **deployment view** of the architecture. The run time processing of the components is represented by these diagrams.
- **Package diagram :** These diagrams show the **organizational units** and their dependencies using their semantics.

Review Questions

1. *What is object oriented analysis and design ?* **AU : May-11, Marks 2**
2. *List various UML diagrams and explain the purpose of each diagram.* **AU : May-14, Marks 16**
3. *How do you see the application of UML diagrams for Iterative Software Development ? Explain.* **AU : Dec.-18, Marks 13**

2.11 Use Case Model **AU : May-11,12,13,14,18,19, Dec.-11,12,13,15,16,17,18, Marks 16**

- During requirement gathering the primary requirements of the system are understood. These basic requirements of the system are called the **functional requirements**.
- In order to understand the working of these functional requirements, the developers and users create a set of scenarios. These scenarios are nothing but the text stories. These text stories are called as **use cases**
- The use cases are the textual representation of particular scenario, whereas the use **case model** is the graphical representation of the scenario. Following table shows the template of the use case -

Use Case template	
Use Case	Write the name of the scenario.
Primary Actor	Specify the role of the entity who interacts with the system.
Goal in Context	Specify the purpose of this use case scenario.
Preconditions	The condition that is to be satisfied before the use case start.

Scenario	Sequence of steps that describe the main scenario.
Triggers	The triggering event due to which the function starts, is described.
Extensions points	The extension points describe the extend relationship.
Exceptions	Describe the exceptional situations that may occur during the execution of the use case.
Priority	Describe the necessity of the system to be implemented.
Open Issues	Certain additional issues that are required for execution of the scenario and that are not specified in the main scenario.

- For example : Following is an example of informal description or text story for withdrawal of money from ATM

  " Customer wants to operate the ATM machine for performing some transactions. For instance customer wants to withdraw the money. He enters the ATM card and then types in the PIN. The system validates the customer. If the customer is the valid customer then the customer is allowed to do further transactions. Otherwise the card will be rejected. The valid customer enters the amount to be withdrawn. It is then checked if withdrawal amount < balance amount. If it is so, then the machine dispenses the desired amount. If there is no further transactions then the card is ejected. Customer collects the cash, statement and card."

Use Case template	
Use Case	ATM system
Primary Actor	Customer
Goal in Context	To monitor all the functionalities required to establish the session
Preconditions	ATM system has to be programmed and as to recognise and validate the PIN.
Trigger	On completion of every activity a beep has to be generated by the system.
Scenario	1. Customer observes the control panel 2. Customer enters the ATM card 3. Customer enters the PIN. 4. Customer selects the operation(withdrawal, deposit, inquiry) 5. Customer collects the cash, statement, card etc.
Exceptions	1. The control panel is not ready. 2. PIN is incorrect. 3. Card is not recognised. 4. Insufficient balance 5. Limit for the transaction exceeds. 6. Total number of allowed transactions per day.

Priority	Essential and must be implemented in banking system.
Secondary Actor	Administrator
Open Issues	1. Is there a need to display any additional information by the control panel ? 2. For how many time the PIN is allowed to enter on incorrect supplement. 3. How much time is allotted to the customer to pick the items like cash or card after completion of operation ?

Key Point : *Use cases are not diagrams they are text.*

2.11.1 Actors, Scenarios and Use Cases

- Use cases are the fundamental units of modeling language in which functionalities are distinctly represented.
- The first step in writing use case is to identify the **actors**. The actors are the entities that use the system or product within the context of **behavior** of the system. Actors **represent the role of** the people as system gets operated. Actors can be a person, computer system or organization.

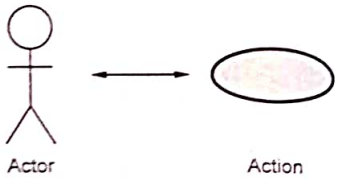


Fig. 2.11.1 Actor and Action representation

- Actor is anything that communicates with the system or product. It is external to the system. Every actor has one or more goals when using the system.
- **Example :** Customer, Inventory database, Accountant
- **Scenario** is a specific sequence of actions or interactions between actor and the system. It is also called as **use case instance**. The scenarios describe the text stories. For example : Scenario of issuing of a book from library, withdrawal of money from ATM.
- **Use case** a success or failure scenarios in which the actor communicates with the system in order to achieve certain functionality.
- For example : Online purchase system

Main scenario : User browses for the product on-line. Then user selects the desired product and adds it in shopping basket. System then displays the shopping basket with newly added product. Now user proceeds for the checkout. If the user is a new customer then it has to register himself as a new register, then this user might be sent with a password via E-mail. User then enters the password and submits the payment and shipping information. The payment is further validated by the system. Finally the system displays the order placed by the user and get it confirmed. The system then confirms the cell.

Alternate scenario : If the credit card information is incorrect, then asks the customer to try again or asks for giving the cash incase.

2.11.2 Use Cases and Use Case Model

- Use case modeling is done during the requirement analysis phase.
- Use cases are the text documents and the use case modeling is an act of writing the text and not the diagrams.
- In requirement analysis phase, along with use case modeling other documents such as glossary, business rules and standards can also be created.
- Use case modeling may include UML diagrams optionally. The use case diagram shows the actors and related functionalities.

Abstract and Concrete use Case

Abstract use case is use case which does not have complete declaration. The declaration in this use case is incomplete. An abstract use case is intended to be used by other use cases, e.g., as a target of generalization relationship.

For example : Refer Fig. 2.11.2 (a).

Use Cases that are self-contained and complete are said to be *concrete* use cases, in that they are actually instantiated by a particular actor(s). (Refer Fig. 2.11.2 (b))

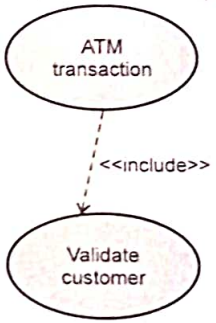


Fig. 2.11.2 (a)

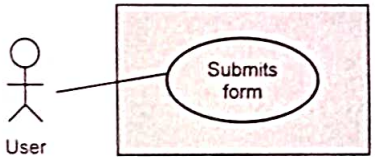


Fig. 2.11.2 (b)

Example 2.11.1 *By considering the library management system, perform the object oriented system development and give the use case model for the same (use include, extend and generalization).* **AU : Dec.-11,12, Marks 16**

Solution :

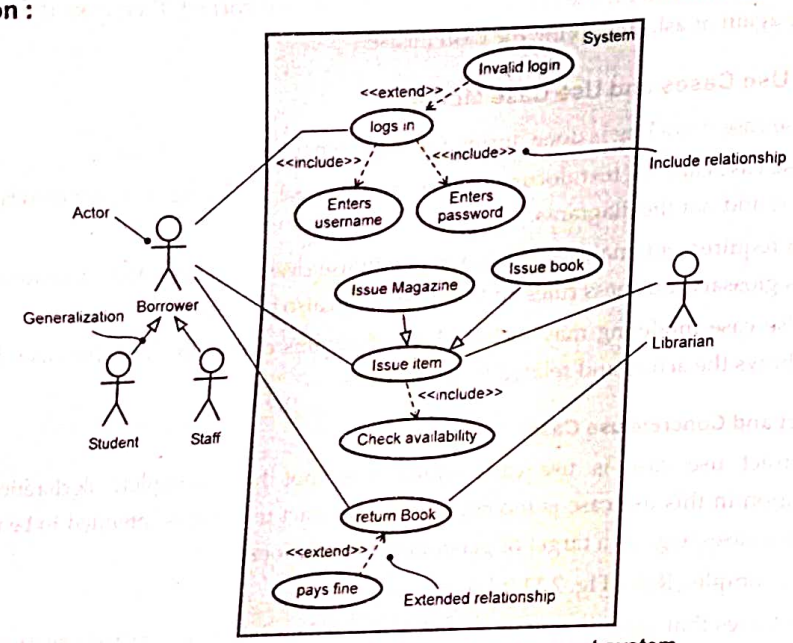


Fig. 2.11.3 Use case diagram for library management system

Example 2.11.2 *Design use case diagram for withdrawal of money from ATM system.*

Solution :

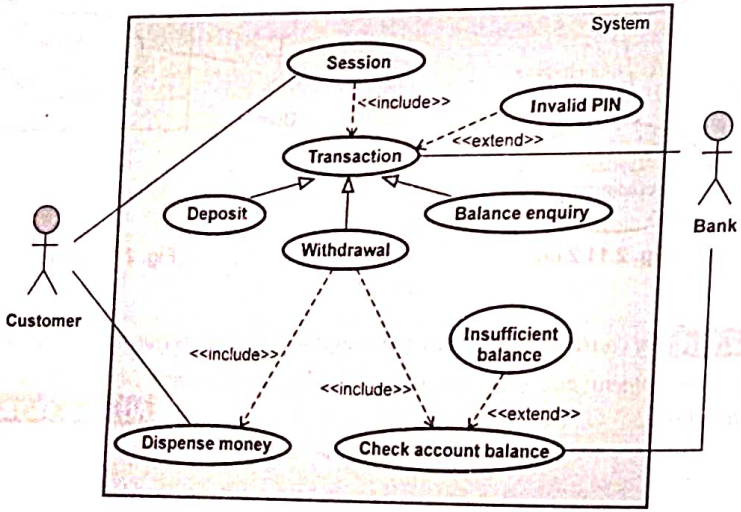


Fig. 2.11.4

Example 2.11.3 *Model a use case diagram for the following scenario : Deepthi Super market wants a subsystem to process the supply orders via the web. The user will supply via a form their name, password, account number and a list of supplies along with an indication of the quantities desired. The system will validate the input, enter the order into a database and generate a receipt with order number, expected ship date and total cost of the order. If the validation step fails, subsystem will generate an error message describing the cause of failure.*

AU : Dec.-17, Marks 8

Solution :

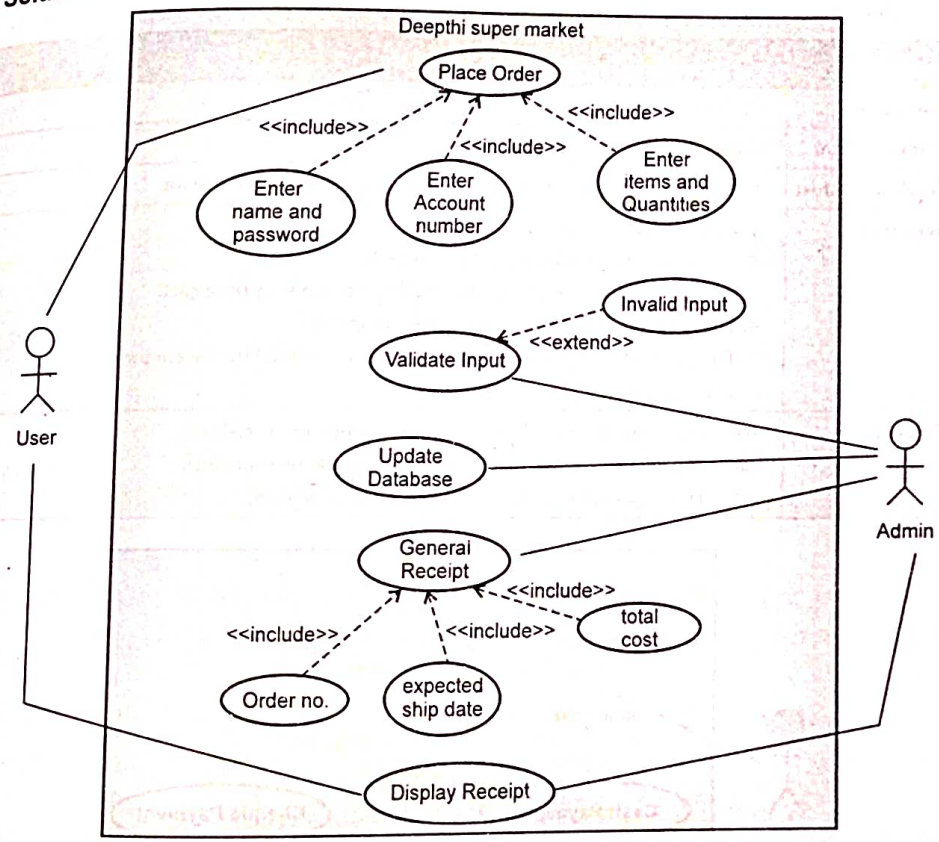


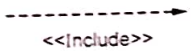
Fig. 2.11.5

2.11.3 Relationships in Use Cases

Several use cases can be related with each other. Similarly the use cases are also related to the actors who are intended to achieve the goal of the system. The use cases are related to any other entity by using include, extend and generalization relationship.

2.11.3.1 Include

- This is the most commonly used relationship which denotes that a given use case may include another. It is denoted by



The use case at the arrow head position is the use case which is included by use case on the other side of the arrow. When there are multiple steps to carry out the single task then this task is divided into the sub-functions and these sub-functions are denoted by the use cases. For example -

Use Case template	
Use Case	On-line shopping
Primary Actor	Customer
Goal in Context	To monitor all the functionalities required to establish the session
Scenario	1. Customer logs in to purchase the product online 2. Customer browses through the products. 3. Customer selects the product and adds it to the shopping cart. 4. The contents of the shopping cart are displayed. 5. The product and shipping information is confirmed by the customer. 6. The customer pays for the product
Extensions	6a The customer pays by credit card (include relationship) 6b The customer pays by the cheque (include relationship) 6c The customer pays by cash (include relationship)

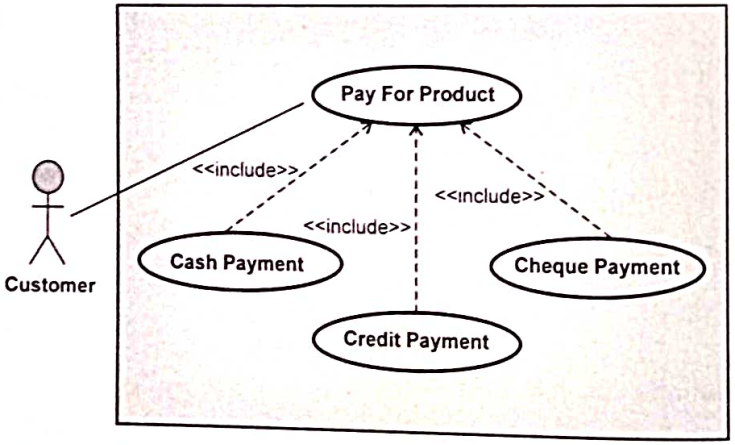


Fig. 2.11.6 Use case for payment for the product (Include-relationship)

2.11.3.2 Extend

- The extend relationship is used when an extended use case is connected to the base use case. While using the extend use the extension points can be explicitly shown in the base use cases. Following are the reasons why extensions occur -

1. When some part of the use case is optional and we need to show separate behavior of the system.

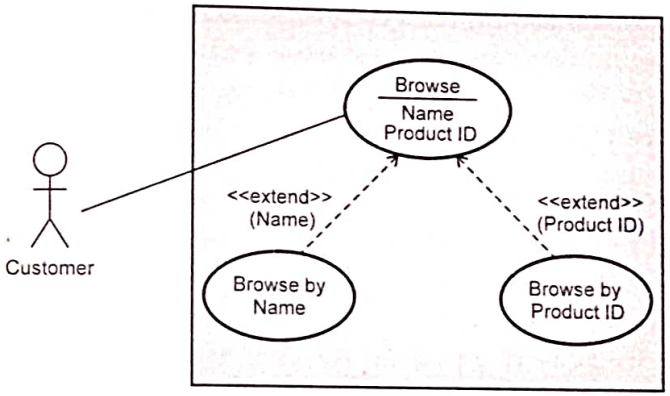
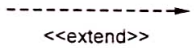


Fig. 2.11.7 Use case for browsing product (Extend - relationship)

2. When we want to show a subflow of the system when some specific conditions occur.

- This relationship is denoted by -



- The extension is conditional. That means its execution is dependent on what has happened while executing the base use case. These conditions are described by the extend relationship.
- The extension use case may access and modify attributes of the base use case
- Example : **Online purchase system,** when user browses for the product, it may browse either by the product name or by product ID.

Use Case template	
Use Case	On-line shopping
Primary Actor	Customer
Goal in Context	To monitor all the functionalities required to establish the session
Scenario	1. Customer logs in to purchase the product online

	2. Customer browses through the products. 3. ... 4. ... 5. The customer pays for the product
Triggers : Extensions points	Customer browses the catalog. 2a Customer browses the product by entering the product name 2b Customer browses the product by entering the product ID.

- Note that extension points are used when we use the extend relationship. The extension points are the labels in the base use case which are referred by the extending use case.

2.11.3.3 Generalization

- In this type of relationship there are two types of use cases - the parent use case and the child use cases.
- A parent use case may be specialized into one or more child use cases that represent more specific forms of the parent. The child use cases inherit all the structure, behavior and relationships of parents.
- Generalization is used when we find two or more use cases having common structure and behavior and purpose.
- The test word used for generalization relationship is "**kind of**". It is denoted by

- For example : For the **Purchase system** the customer can place an order via telephone or using internet. The use case

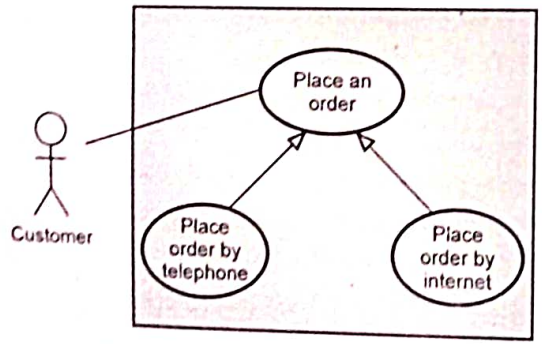


Fig. 2.11.8 Generalization Use case

2.11.4 When to Use Use Cases ?

- Use cases are used to model the behavior of the system. Following are some reasons why use cases are important -
  1. By modeling the **behavior of elements** using use cases will provide the outside view using which the inside view of the system can be constructed.
  2. Using use cases, developers can identify and **understand various system elements**.
  3. Use cases serve as the **basis for the test cases** that can be carried out during system development.
- Following are the steps that are used to model the behavior of the system -
  - Identify the actors that interact with the system. Actors interact with the system which is intended to perform some task.
  - Organize the actors by identifying the generalized and specific roles.
  - There are various ways by which the actor interacts with the system. There are some interactions in which the interactions change the state of the element. Some interactions occur when elements response to the event.
  - There are some exceptional ways in which each actor interact with the element.
  - For factoring the common behavior, make use of include relationship and for exceptional behavior make use of exclude relationship.
- Following is an example of the hospital management system in which the receptionist interacts with the hospital management for making admission to the hospital.

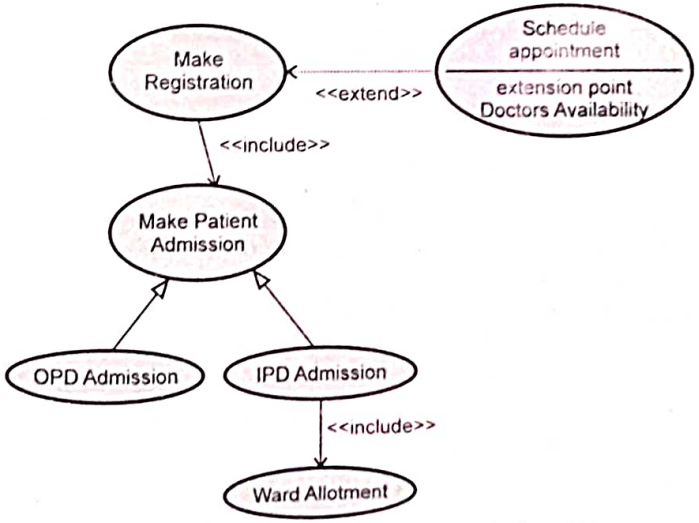


Fig. 2.11.9 Use case for admission to the hospital

Review Questions

1. *Explain with an example, how use case modeling is used to describe functional requirements. Identify the actors, scenarios and use cases for example.* **AU : May-11,12,13,14, Dec.-13,16, Marks 16**
2. *Explain about Use-Case Model for a case study of your choice.* **AU : Dec.-15, Marks 16**
3. *Explain with an example a concrete use case and an abstract use case.* **AU : Dec.-17, Marks 5**
4. *Explain in detail about use case diagrams.* **AU : May-18, Marks 6**
5. *Explain briefly the elements of Use Case diagram.* **AU : Dec.-18, Marks 13**
6. *Write the use of case modeling and explain in detail.* **AU : May-19, Marks 7**

2.12 Class Diagrams AU : May-11,12,16,17,18,19, Dec.-11,12,13,14,15,16,17,18, Marks 16

A class represents a collection of objects having the same characteristic properties that exhibit common behaviour. It gives the blueprint or description of the objects that can be created from it.

- There are **three strategies** that can be adopted for finding the conceptual classes -
  1. Reuse or modify existing models for finding the conceptual classes. This is the easiest way because in the existing model the domain models and data models are well designed. The common domains can be used to identify the conceptual classes.
  2. Use of category list.
  3. Finding out the name phrases from the problem description. The name phrases denote the conceptual classes.
- Following are the **steps to create a domain model used** for representing the conceptual classes.
  1. Find out the conceptual classes using the above described three strategies.
  2. Draw the UML class diagram for these classes.
  3. Add associations and attributes to these classes.
- Conceptual classes can be identified using the **category list**. Some commonly used categories are as shown in the following table -

Category	Example
Physical Objects	Ticket, Die, Board, Register, Item, Book.
Place of service	Store, Airport,ATM.

Catalog	ProductCatalog, BookCatalog.
Role	Passanger, Accountant, Student, Customer, Librarian, Administrator.
Financial Items	Credit Card, Cash
Schedules	DailySchedule, RepairSchedule.
Record keeping	Database, Register, Ledger.

- The conceptual classes can be identified by identifying the **noun phrases**. This technique is suggested by Abbot. This technique is called **linguistic analysis.**

Example : ATM System

Problem Description : Consider an ATM System in which the customer with an ATM card can establish session with the system. First of all he inserts the card into the system and enters the PIN. The console of the system reads the card and authenticates the customer. The system has console containing display and keyboard.

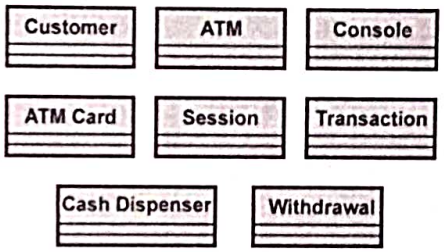


Fig. 2.12.1 Initial ATM system

The session with the system can be accomplished by the transaction. Using the withdrawal transaction the customer can get the requested amount through cash dispenser.

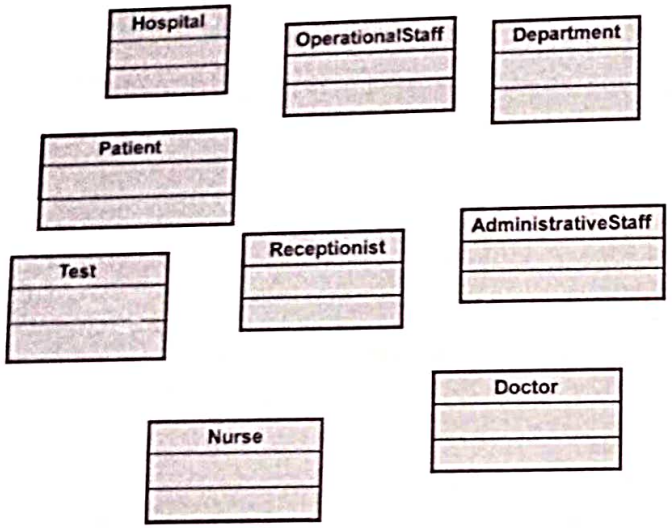
From above problem description, we can obtain the conceptual classes. The conceptual classes can be identified by identifying the noun phrases.

The UML class diagram using these conceptual classes can be - as shown in Fig. 2.12.1.

The associations and attributes for the above class diagram is not shown.

Example 2.12.1 *For the Hospital Management system draw conceptual class diagram(overall system)* **AU : Dec.-12, Marks 8**

Solution :



2.12.1 Attributes

Attribute is a logical data value of an object. It is necessary to identify the attributes of conceptual classes so that information required by the system can be represented.

The attributes are needed when some specific information needs to be remembered. For example -

- **Customer** needs **ID**
- **Account** has account type
- **Flight** has **Date/time**
- **Student** has **name and address**

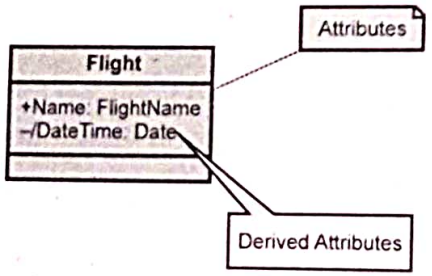


Fig. 2.12.2 Attributes and derived attributes

2.12.2 Relationships

Various relationships in class diagram are –

1. Associations
2. Inheritance
3. Aggregation
4. Composition

2.12.2.1 Associations

- The association is the most common type of relationship used UML. The association is shown between the classes or instance of classes. For example -

In above Fig. 2.12.3 the relationship between the customer and the bank account is shown by the association relationship. This can be read as "**One Customer can have multiple Accounts in the bank**".

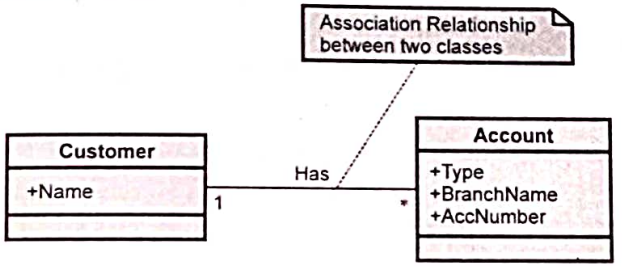


Fig. 2.12.3 Association relationship

- **When to show an Association?**

The association relationship is used in the system design when we want that the relation between two object has to remembered for some time.

The association relationship can be defined from the common association list. Use of too many association relationships in the class diagram must be avoided.

- **How to represent Association in UML?**

The association is represented by a line between the classes. The **association name** must be denoted by the string with first letter capital.

The ends of association must represent the multiplicity expression indicating the numerical relation between the classes.

- Using the simple association names like **Has** or **Uses** leads to less understanding of the domain model. Hence more appropriate verbs can be used to denote the association name. Most commonly used association names are -

1. A uses B
2. A has B
3. A is a member of B
4. A records B
5. A issues/returns B
6. A communicates with B
7. A is followed by B
8. A is owned by B

- **Example**

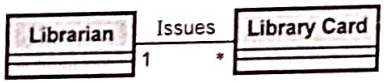


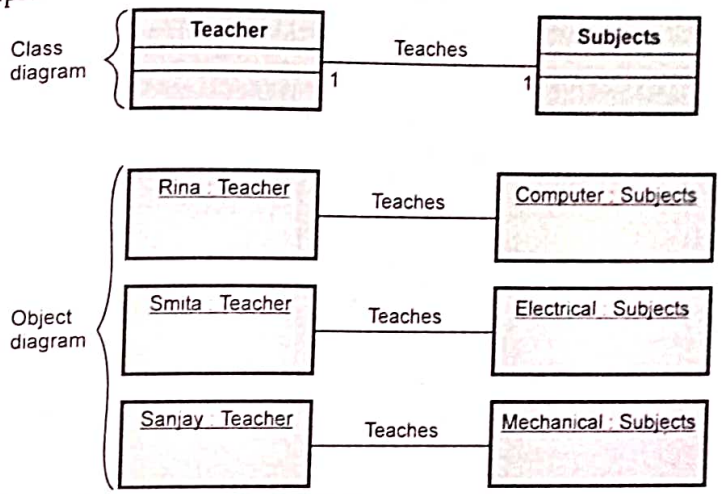
Fig. 2.12.4

- **What is multiplicity in an association?**
  - Multiple objects can be related to some objects. The multiplicity represents "how many" objects are connected.
  - The multiplicity can be written as expression. It describes "one" or "many" objects associated with other object.
  - The multiplicity is specified at the end of the association link.
  - Following are the notations that can be used to denote the multiplicity of associations -

Notations	Description
1	Only one instance
0..1	Zero or one instance
*	Many instances
0..*	Zero or many instances
1..*	One or many instances
2..10	You can specify the integer range

  - There is a difference between two terms - the **multiplicity** and **cardinality**. The multiplicity is the constraint on the collection of the associated **objects** whereas the cardinality is the count of the objects that are in collection. The multiplicity is actually the constraint on the cardinality.
  - While making the class model first decide about the classes and their associations and then think about their multiplicity.

For example -



- **How to find association with common list?**

Following table shows different associations that can be identified from the common association list -

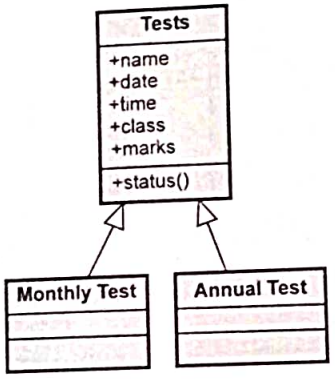
Category	Example
A and B are transactions and they are related with each other	**Reservation** is related to the **Confirmation** of Reservation
A is a role and B is transaction	**Customer** makes the **Payment** **Librarian** issues **Book** **Librarian** issues **Library Card**
A is logical or physical part of B	**Customer** has **Library Card**
A is logically or physically contained within B	**Book** is present in **Shelf**
A is a member of B	**Student** is a member of **Library**
A uses/manages/owes B	**Customer** uses **Account**

2.12.2.2 Inheritance

- Inheritance or **Generalization** is a relationship between superclass and one or more subclasses.
- This is a kind of relationship in which the properties of parent class are inherited by the child class.
- In this relationship the **superclass** is more **general** and the subclasses are more specific. The subclasses use the common properties and methods of the superclass but at the same time adds specific attributes, operations and associations.
- This is **is-a** kind of relationship because each instance of subclass is an instance of superclass.

- The inheritance or generalization is represented by an hollow arrow head as given below -

- For example



2.12.3 Aggregation and Composition

Aggregation and composition are the kind of associations.

Aggregation

Aggregation is used to represent the whole-part relationship. It normally posses the Has-a relationship. It is denoted as follows -

Fig. 2.12.5 Aggregation relationship

For example - DVD player and Car are the two classes that can be associated by the aggregate relationship. Note that the DVD player can exist without Car and Car can exist without the DVD player. We can read the relationship as "DVD player is a part of Car".

Fig. 2.12.6 Example - Aggregate relationship

Composition : Composition is a special kind of aggregation which represents the whole-part relationship. To be more specific, a **restricted** aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. It is denoted as follows -

Fig. 2.12.7 Composition relationship

For example - Steering and Car are the two classes those can be associated by composite relationship. This is restricted relationship because the Car can not exist without Steering.

We can read the relationship as "Steering is a part of Car"

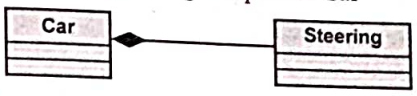
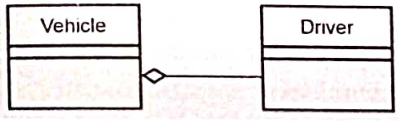
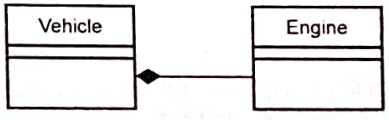


Fig. 2.12.8 Example - Composite relationship

Difference between aggregation and composition

Aggregation	Composition
The relationship between child and parent class is represented in aggregation relationship where in child can exist independently of the parent.	The relationship between child and parent class is represented in composition relationship where in child can not exist independently of the parent.
It is weak association.	It is strong association.
Deletion of owning class doesn't affect system.	If owning class is deleted it will significantly affect the system.
The hollow diamond is used in arrow notation.	The solid diamond is used in arrow notation.
For example 	For example 

2.12.4 Visibility

The visibility specifies the how the attributes and operations are visible in the system. There are four levels of visibility -

1. **public :** The public element is visible to all elements that can access the classifier. It is denoted by '+' symbol.
2. **protected :** The protected element is visible to all the elements that have a generalization relationship to the namespace that owns it. **Protected** visibility is represented by '#' symbol.

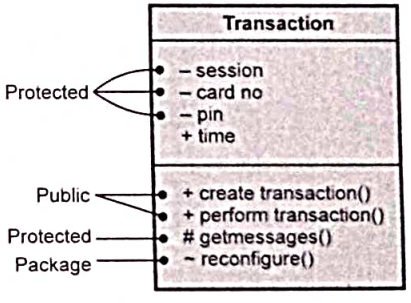


Fig. 2.12.9 Visibility

3. **private :** A private element is only visible inside the namespace that owns it. Private visibility is represented by '-' symbol.
4. **package :** Only the elements that are declared in the same package can use this feature. It is specified using the symbol '~'.

2.12.5 Qualified Association

The qualified association has **qualifier** which is used to select particular object from the set of objects. For example For selection of particular **Student** from the **Course** the qualifier will be **StudentID**

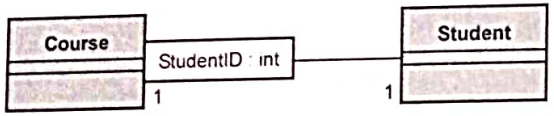


Fig. 2.12.10

A qualifier is a property which defines **selection key.** UML allows having a qualifier on each end of a single association. The multiplicity can be associated with the association link.

2.12.6 Association Class

The abstract class is a class that allows the association to be a class itself. That means when two classes are related with each other by an association link, then the association itself can have attributes and operations. Hence this association can be represented by a class. For example- in the following illustration the **Enrollment** is an association class for the classes **Student** and **Course.**

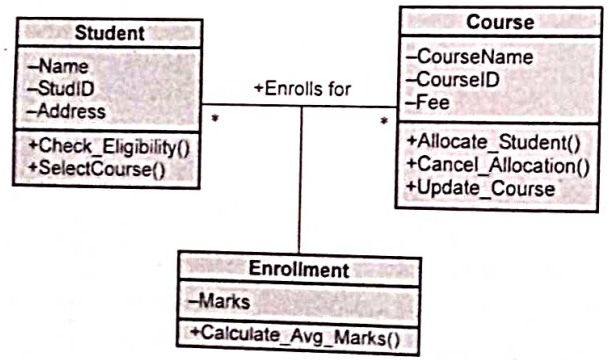


Fig. 2.12.11 Association class

2.12.7 Singleton Class

When there is a situation that only one instance of a class is instantiated then it is called singleton instance. In the UML class diagram such a class is marked with "1" at the upper right corner in the name compartment. Using singleton pattern exactly one object is needed to co-ordinate actions across the system. This concept restricts the instantiation to a certain number of objects.

For example -

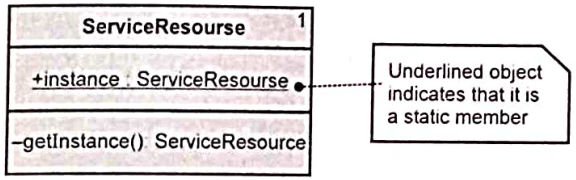


Fig. 2.12.12 Singleton class

2.12.8 Active Class

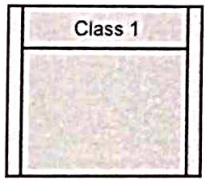


Fig. 2.12.13 Active Class

An Active Class indicates that, when instantiated, the Class controls its own execution. Rather than being invoked or activated by other objects, it can operate standalone and define its own thread of behavior. In other words this class initiates a thread or a process. For example - CommandProcessor is an active class when user submits some command then based on that command particular process is invoked for execution.

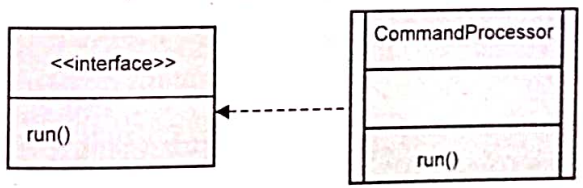


Fig. 2.12.14

2.12.9 When to Use Class Diagrams?

Following are uses of Class diagram :

1. **For modeling the vocabulary of the system the class diagram is used.**
   - When we want to decide which abstractions must be the part of the system and which abstractions should be outside the boundary, we design the class diagram.
   - The class diagram is used to represent the abstractions and their responsibilities.
2. **For modeling the collaborations of the system the class diagram is used.**
   - Collaborations is a collection of classes, interfaces and other elements that work together to provide some co-operative behavior.
   - The class diagram specifies such collaborative classes and their relationships.
   - For example -

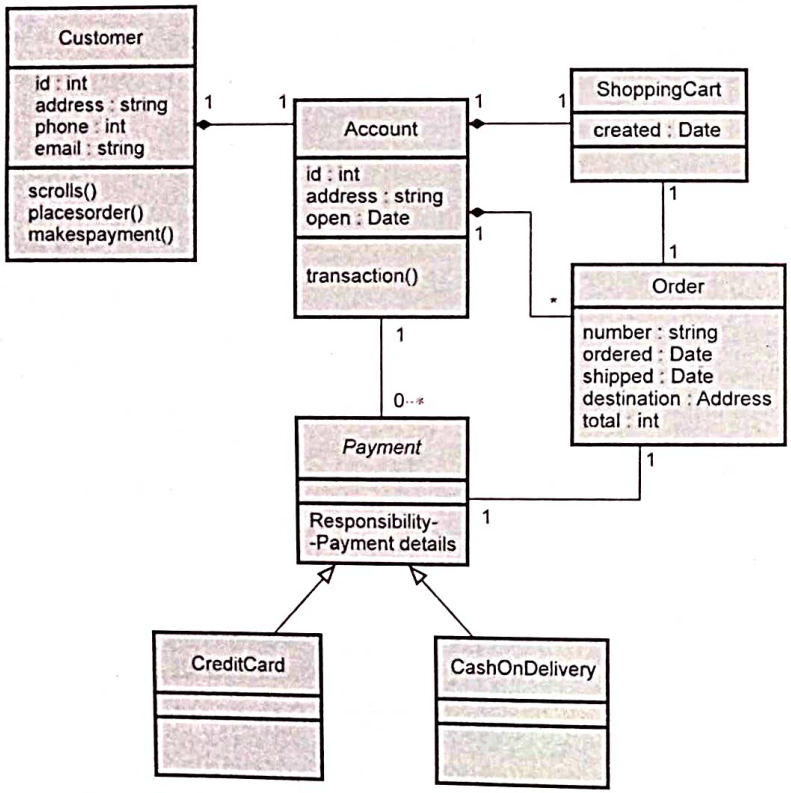


Fig. 2.12.15 Modeling simple collaborations for online shopping

3. **For modeling the logical database schema the class diagram is used.**
   - For conceptual design of the databases the schemas are used.
   - For example -

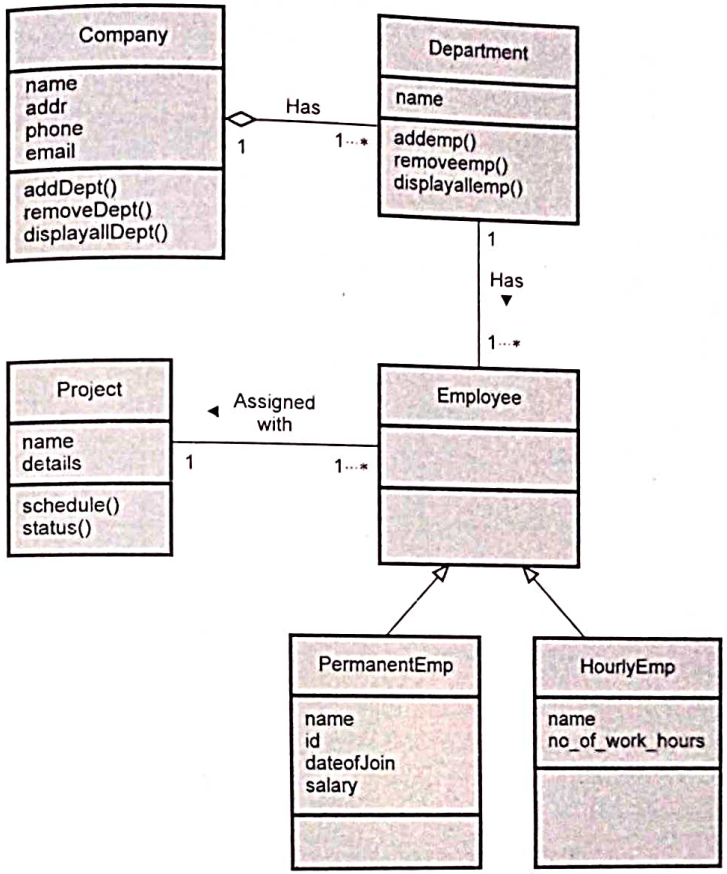


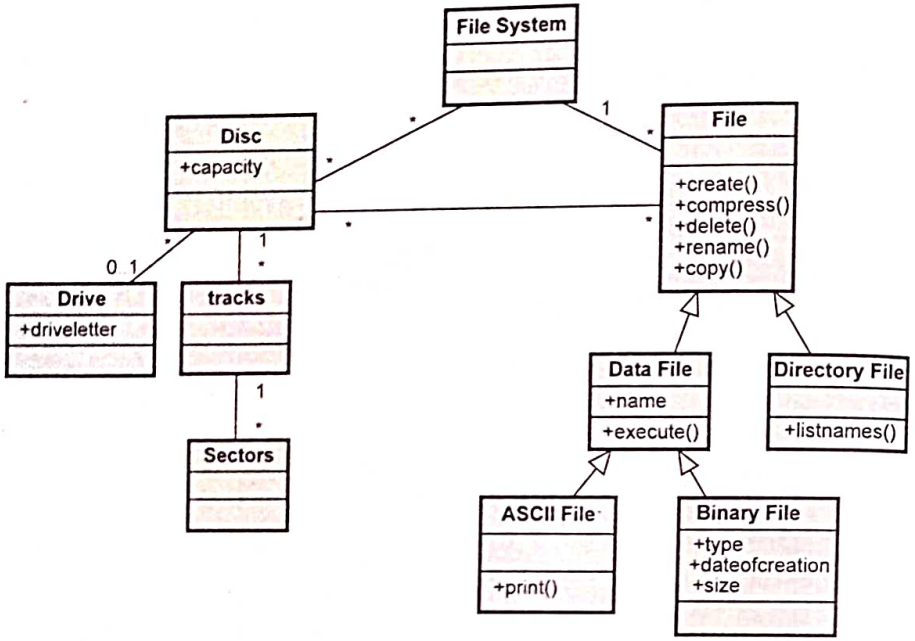
Fig. 2.12.16 Company Information System

2.12.10 Examples on Class Diagram

Example 2.12.2 *Prepare a class diagram for each group of classes. Add at least 10 relationships (association and generalizations) to each diagram.*

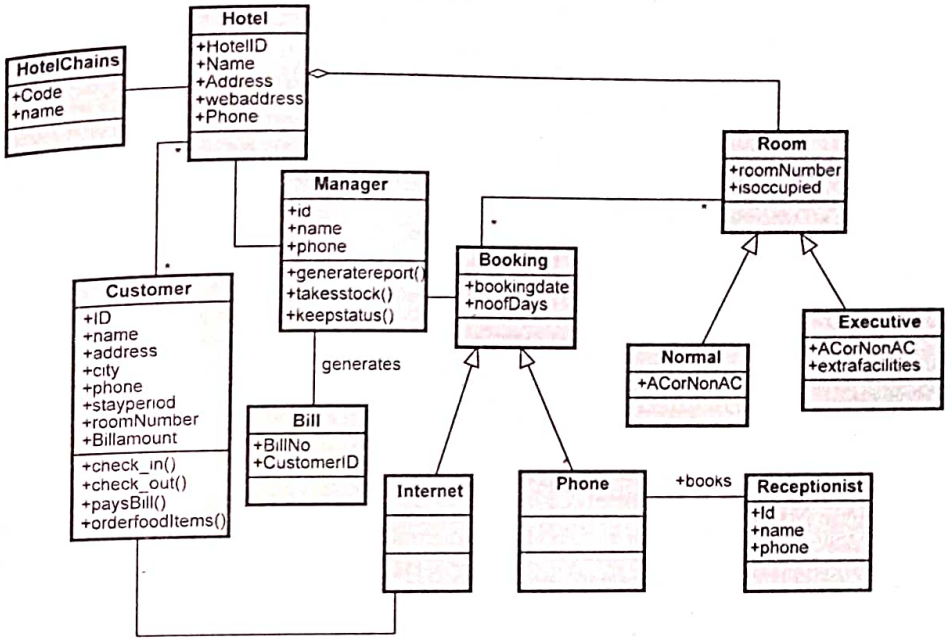
File system, file, ASCII file, binary file, directory file, disc, drive, track, sector

Solution : The required class diagram is given below -



Example 2.12.3 *Prepare a class model for the hotel management system. The system should support chain of hotels. A hotel contains two categories of rooms : executive and normal, both AC and non-AC. The customers of executive rooms can avail extra facilities like games, swimming, food service in rooms, etc. The booking is possible by Internet or by phone. If the booking is through phone, process is done by receptionist and if booking is done through Internet the process is carried out by customer through hotel website. Depending on the number of days customer stays, appropriate bill is generated. The bill also contains amount for transport, food and other facilities enjoyed by the customer along with necessary taxes. The manager should be able to generate reports like list of customers staying in the hotel, list of rooms empty, monthly/yearly income, etc.*

Solution :



Example 2.12.4 *Convert the following into a class diagram with appropriate classes, relationships, multiplicity.*

A computer program has many statements. An expression is a statement. A function is statement. An expression contains a variable, a constant and operator. A relational operator is an operator. An arithmetic operator is an operator. A function has an argument list, a return type.

Solution :

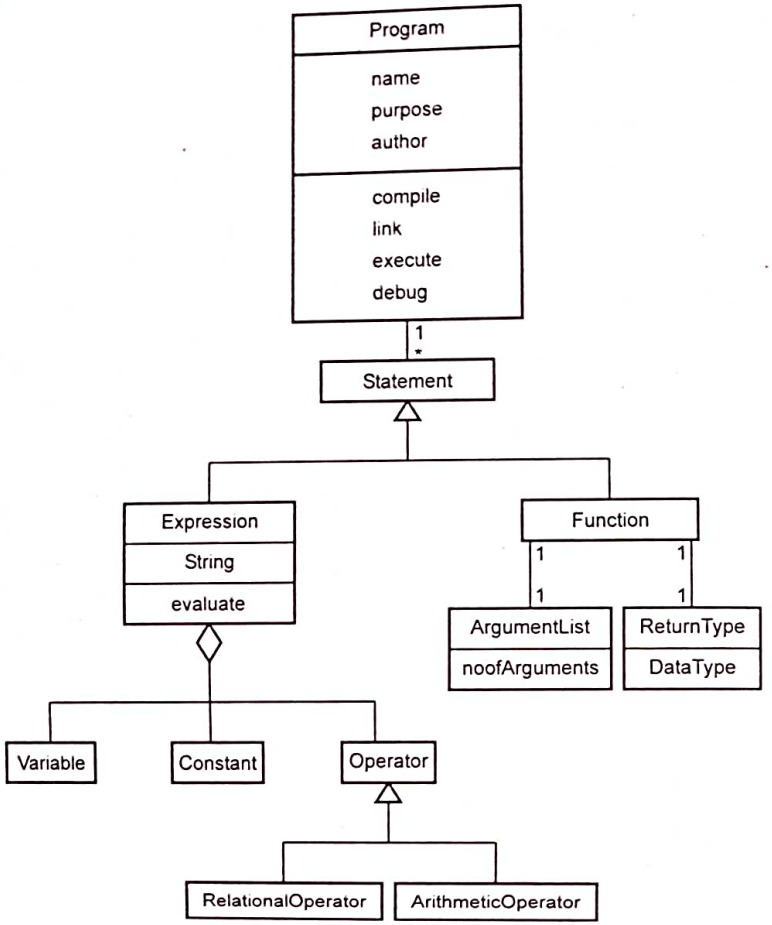


Fig. 2.12.17

Example 2.12.5 *Convert the following into a class diagram with appropriate classes, relationships, multiplicity.*

Car dealer sales cars. A car is owned by an owner. The owner has an address. The owner can be a person, a company or a bank. A car loan may be involved in the purchase of a car. Bank provides the loan.

Solution :

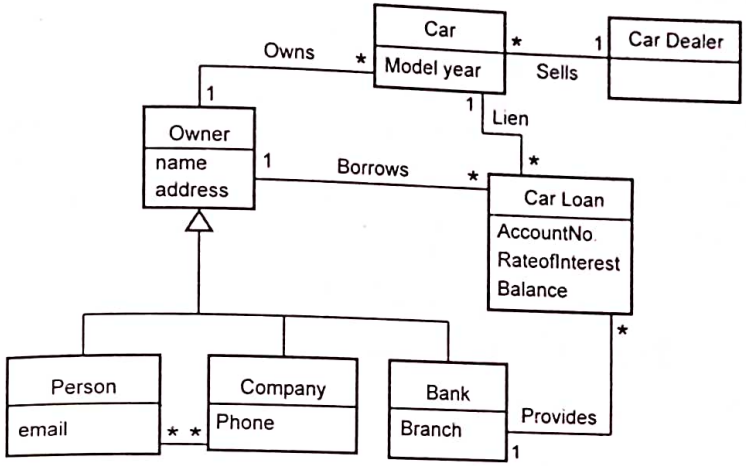


Fig. 2.12.18 Class diagram

Example 2.12.6 *Convert the following into a class diagram with appropriate classes, relationships, multiplicity.*

A drawing object is a text, a geometric object or a group. Group can contain at least two geometric objects. Point, polygon are geometrical objects. A polygon is composed of an ordered set of points.

Solution :

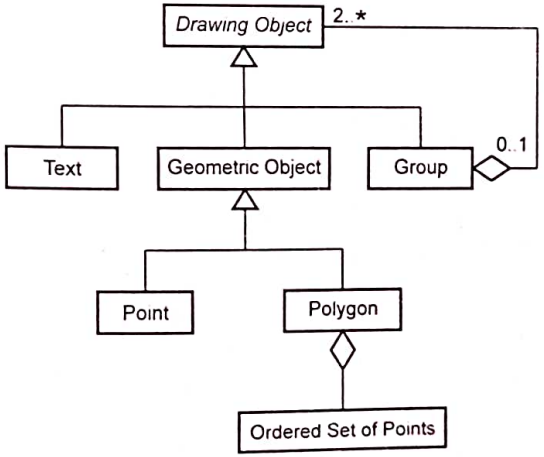


Fig. 2.12.19

Example 2.12.7 *Draw a class diagram for the given description with appropriate relationships, relationship names and multiplicity.*

A person owns multiple documents. Each document is composed of paragraphs that are, in turn, composed of characters. The copy operation propagates from document to paragraph to characters. Copying a paragraph copies all the characters in it. The operation does not propagate in reverse direction; a paragraph can be copied without copying the whole document.

Solution :

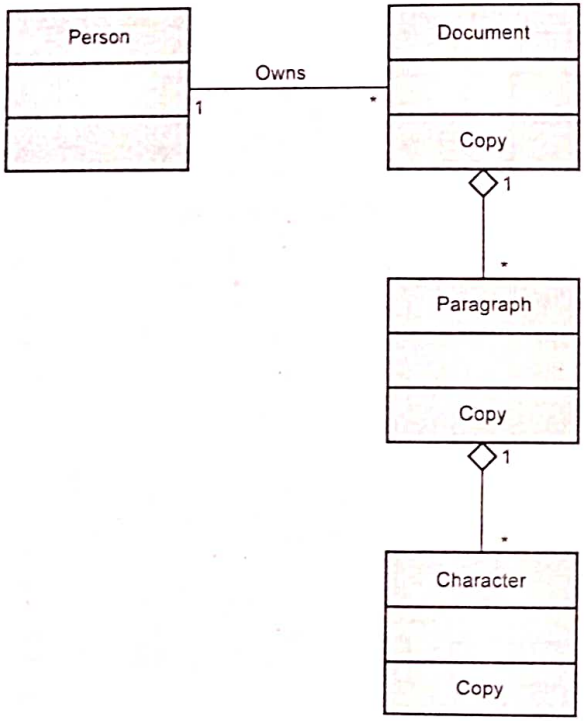


Fig. 2.12.20 Class diagram representing propagation operation

Example 2.12.8 *A project has three to five students. A projects has a guide can guide one to three projects. For this description draw a class diagram. From the class diagram draw an object to show two projects, seven students and one guide. Do not write any explanation, just draw the diagrams.*

Solution :

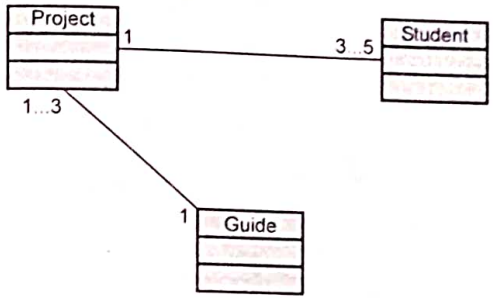


Fig. 2.12.21 (a) Class diagram

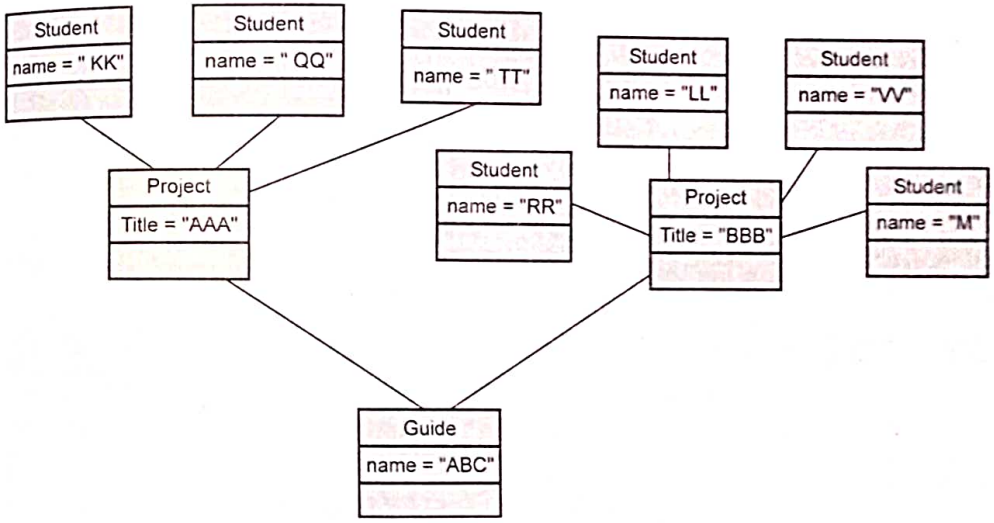


Fig. 2.12.21 (b) Object diagram

Example 2.12.9 *Convert the following description into a class diagram. Show classes, relationships and multiplicities.*

In a university there are different classrooms, offices and departments. A department has a name and it contains many offices. A person working at the university has a unique ID and can be a professor or an employee. A professor can be an associate or assistant professor and he/she is enrolled in one department. Offices and classrooms have a number ID, and a classroom has a number of seats. Every employee works in an office.

Solution :

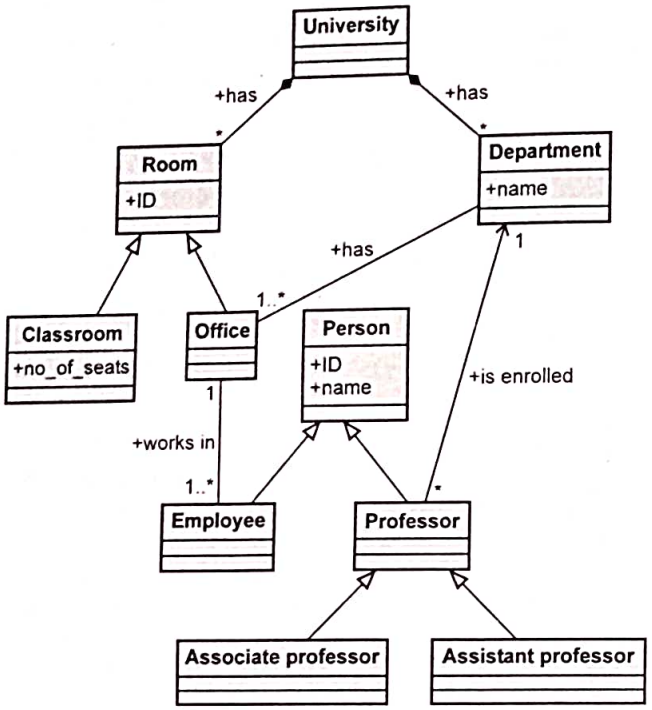
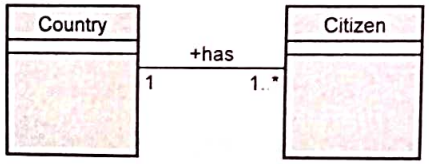


Fig. 2.12.22

Example 2.12.10 *A country has many citizens. One citizen among all can be identified by an Adhar card. Represent this as a qualified association. Two diagrams are expected. One without qualifier and the other with qualifier.*

Solution :



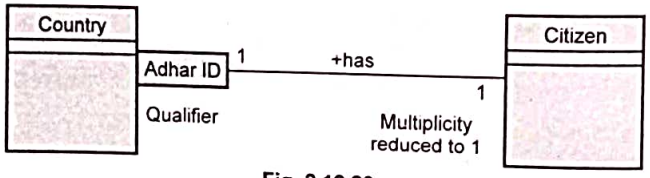
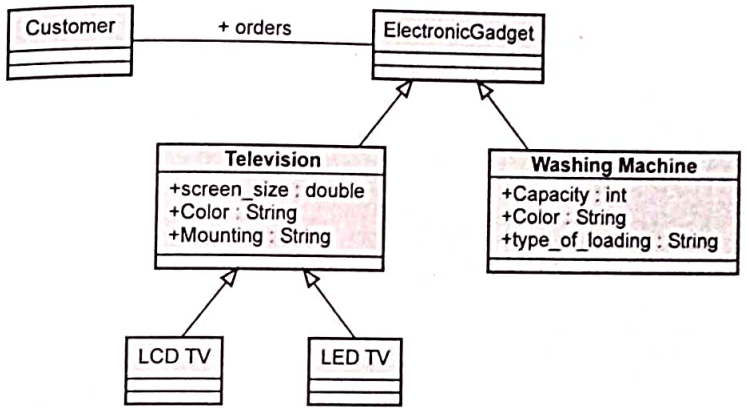


Fig. 2.12.23

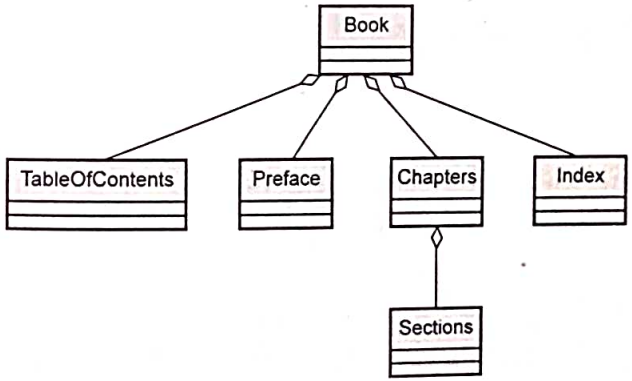
Example 2.12.11 *An electronic gadget shop has television and washing machines. Television has screen size inches, color, mounting(only wall or table mounts are possible). Televisions come in two types namely LCD and LED. A washing machine has a capacity in litres, color, type of loading(Top or front). A customer can order an electronic gadget. Draw a class diagram for this with attributes and relationships.*

Solution :



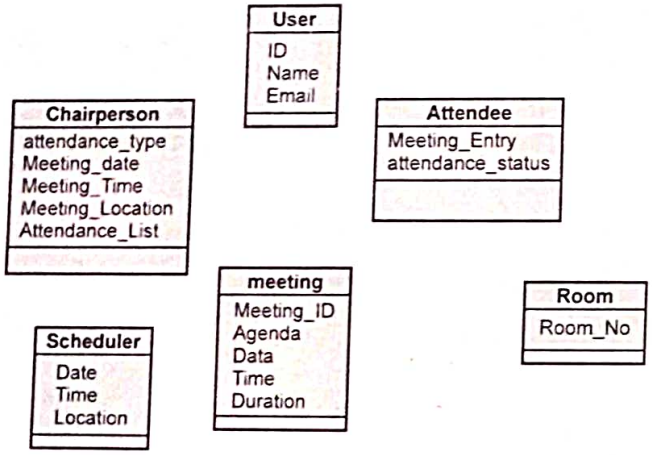
Example 2.12.12 *A book is composed of Table of Contents, Preface, many chapters and index. Chapters are composed of sections. Convert this using a class diagram.*

Solution :



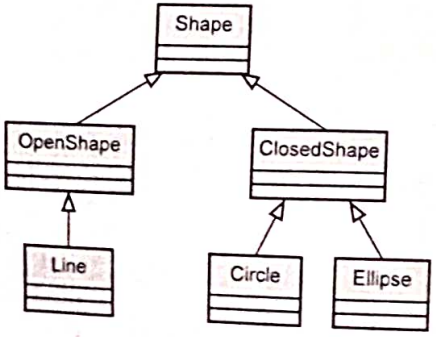
Example 2.12.13 *Analyze the following description and apply steps to find the candidate analysis classes and attributes from it. List all classes and all attributes separately. The scheduling software facilitates the meetings. When a user(chairperson) arranges a meeting, the software places a meeting entry in the schedule of each attendee. The chairperson uses the scheduler to reserve a room for the meeting, to identify attendees, and to find time on their schedules when everyone is available. The chairperson can indicate whether the attendance for each attendee is required or optional. The system tracks the attendance status for each attendee-whether an attendee has accepted or declined.*

Solution :



Example 2.12.14 *Draw a class diagram for generalization relationship using the keywords Shape, ClosedShape, OpenShape, Line, Circle, Ellipse.*

Solution :



Example 2.12.15 *Construct design for library information system which comprises and following notations (i) Aggregation (ii) Composition (iii) Association*

AU : Dec.-15, Marks 16

Solution :

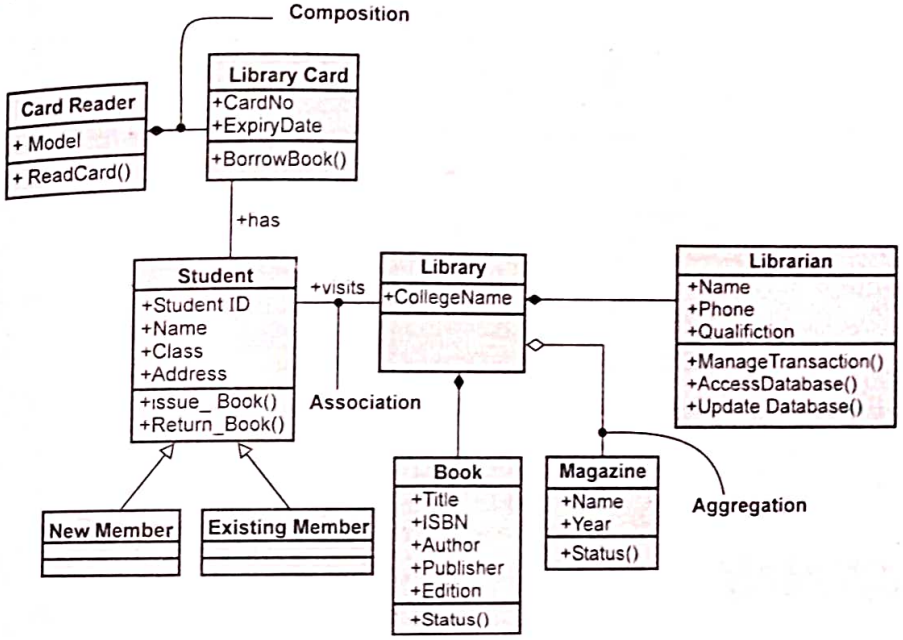


Fig. 2.12.24

Example 2.12.16 *Model a class diagram for "Banking System" State the functional requirements you are considering.* **AU : Dec.-17,18, Marks 15; Dec.-18, Marks 13**

Solution : See Fig. 2.12.25 on next page.

Functional Requirements -

1) The valid customer must log in to the system using username and password.
2) The system should authenticate the user.
3) The system should allow the transfer the funds between authorised access.
4) The system should allow the customer to view the balance in his account.
5) An administrator is that person who is allowed for some editing for the internet banking system like add/cancel customer, check the transactions.

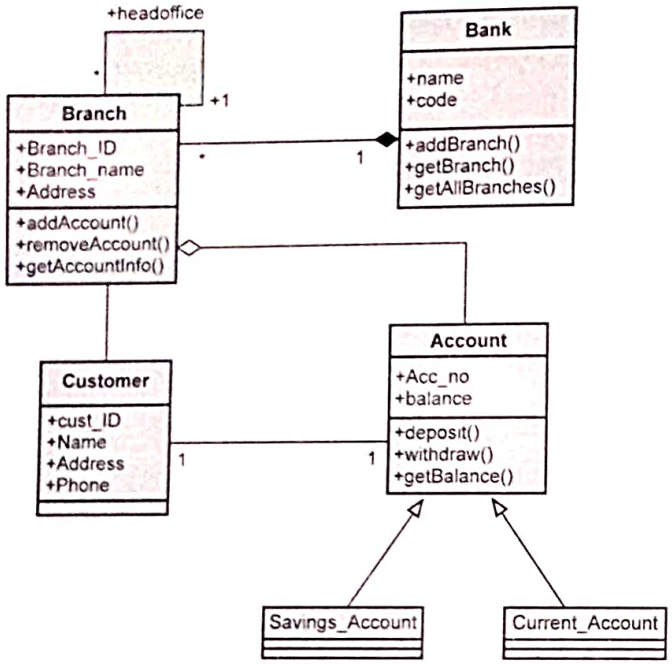


Fig. 2.12.25

Review Questions

1. *Describe the strategies used to identify the conceptual classes. Describe the steps to create a domain model used for representing the conceptual classes.* **AU : May-11,12,18, Dec.-13, 14, Marks 16**
2. *Explain the following with an example : Conceptual class diagram.* **AU : Dec.-11, Marks 8**
3. *Explain the guidelines for finding conceptual classes with neat diagrams.* **AU : May-16, Marks 8, May-17 Marks 10**
4. *Discuss in detail about the three strategies to find conceptual classes.* **AU : Dec.-16, Marks 8**
5. *Explain in detail about conceptual classes and description classes.* **May-19, Marks 5**
6. *What is multiplicity of an association ? Explain with an example the different types of multiplicities.* **AU : Dec.-17, Marks 7**
7. *Write a short note on the associations.* **AU : May-19, Marks 3**
8. *Define aggregation and composition* **AU : May-11, Marks 2**
9. *Explain association, aggregation and composition relationship in detail.* **AU : Dec.-16, Marks 8**
10. *Explain about Aggregation and Composition with examples.* **AU : May-17, Marks 6**
11. *Explain with an example aggregation and composition.* **AU : Dec.-17, Marks 6**
12. *Discuss about aggregation and composition.* **AU : May-18, Marks 7**
13. *Differentiate aggregation and composition.* **AU : May-19, Marks 8**

2.13 Interaction Diagrams

AU : May-11,14,18,19, Dec.-13,14,15,16,17,18, Marks 16

Interaction diagrams are created to represent interactive behavior of the elements of the system.

UML defines two types of interaction diagrams - **1. Sequence diagram** and **2. Communication or collaboration diagram.**

2.13.1 System Sequence Diagram

Definition : In UML, system sequence diagram is represented as the interactions made by the actors with the objects and operations initiated by them.

- The external input events are called the system events which are useful in analyzing the behavior of the system. System sequence diagram are useful UML notations for depicting the behavior of the system.
- The time based events or messages can also be represented by the sequence diagram.
- The sequence diagrams are useful tool to represent **what the system does and not how it does.** The sequence diagram is used as a tool to represent the system as a **black box.**
- Sequence diagrams are derived from the use cases and they represent one scenario.
- Sequence diagram is used to discover software architecture and interfaces.

Various elements of sequence diagram are -

Time line or Life line - The vertical line is called object's life line during the interaction.

Messages - The messages are represented by the line between the two objects. The messages will occur from top to bottom. Each message is labeled. This labeling can be an event that leads some action, self call, control information, condition checking and so on. It can be denoted as

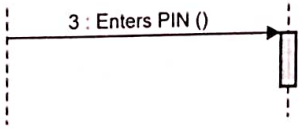


Fig. 2.13.1

Box - The boxes at the top of each time line generally represents the classes, objects or actors. Objects have labels in the standard UML format **name:ClassName,** where **"name"** is optional (objects that haven't been given a name on the diagram are called **anonymous objects**). Classes have labels in the format ClassName and actors have names in the format ActorName.

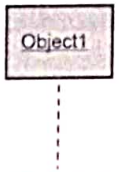


Fig. 2.13.2

Activation Box - The long thin boxes on the life line are called the activation boxes.These are also called as **method invocation boxes**. These boxes indicate that the processing is being done by the target object to fulfill a message. It is denoted as -

Fig. 2.13.3

X at the bottom - The cross at the bottom indicate that the particular object can be removed from the memory. It is represented as

Fig. 2.13.4

Conditional logic - The model can show the conditional logic by using the conditional messages.

The format of the sequence diagram will be like this -

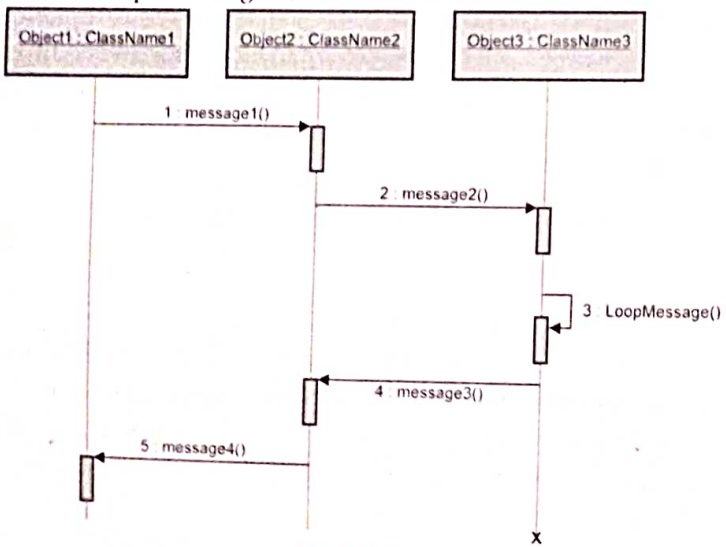


Fig. 2.13.5

2.13.2 Relationship between Sequence Diagram and Use Cases

- Sequence diagram is used to represent one scenario of the use case. Hence it can be derived by observing the use cases.

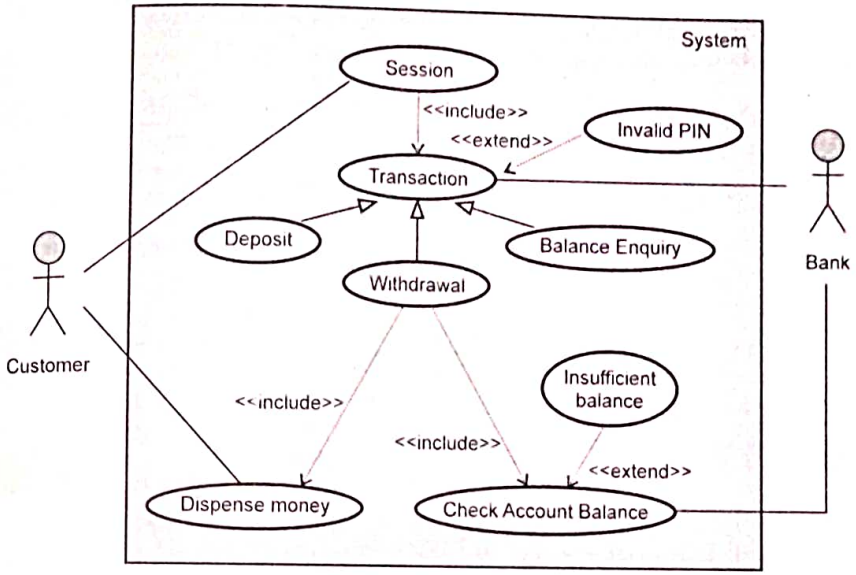


Fig. 2.13.6 Use case diagram for withdrawal of money

- Following is a use case diagram for the withdrawal of money from ATM system.
- Following is a system sequence diagram for **Withdrawal of money from ATM system.**
- One scenario of the use case can be represented using the system sequence diagram. The loops in the system can be shown using the **interaction frame.**

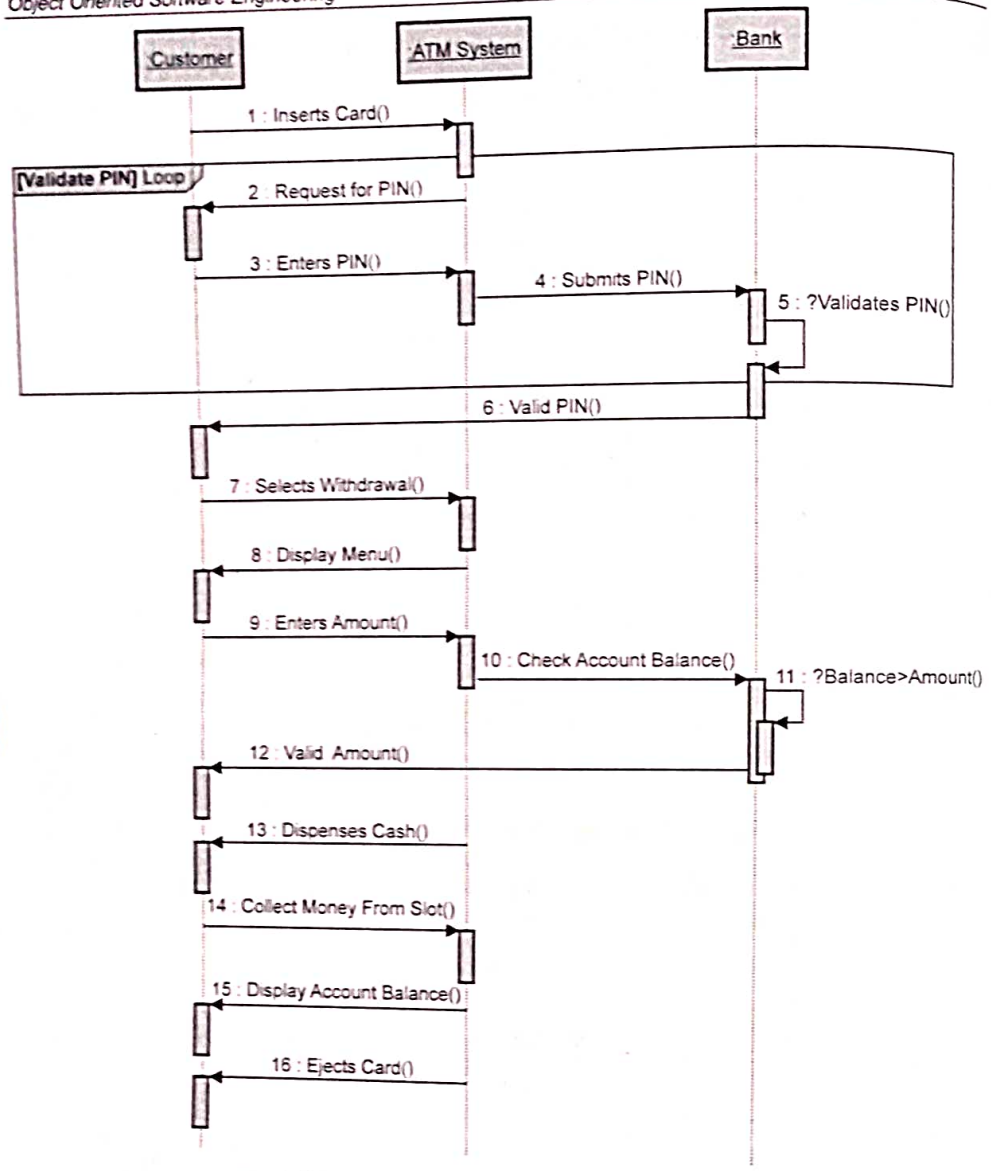


Fig. 2.13.7 Sequence diagram for withdrawal of money

Guideline

Following guideline is used while designing the SSD

1. The Sequence diagram generally represents only **one scenario of use case** model.

2. System events must be represented at the abstract level instead of representing in terms of physical input device.
3. The system events must be usually a verb. For example - display, enter, select, eject and so on.
4. The sequence diagram can represent the collaboration with other external systems. For example - For representing the scenario **withdrawal of money** the sequence diagram can be drawn for the objects **Customer** and the **ATM System.** The **Bank** can be external system in this model and the interaction with such an external system can also be represented using sequence diagram.

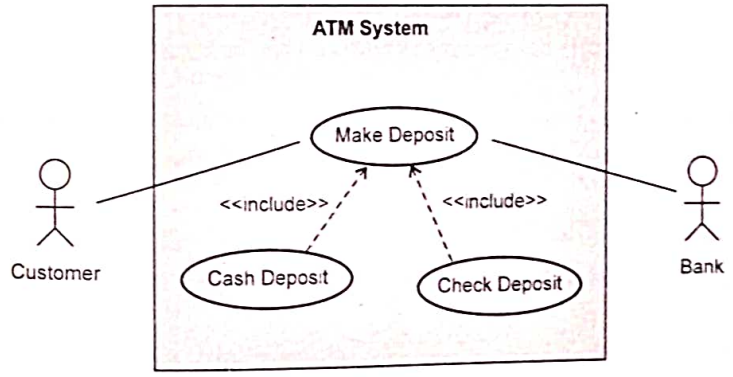
Example 2.13.1 *Consider the following use cases that play a role in a banking system :*

i) Deposit

ii) Withdraw (Minimum balance has to be checked)

Model sequence diagram for the above two cases **AU : Dec.-17, Marks 13**

Solution : i) Deposit



ii) Withdraw - Refer Fig. 2.13.6.

Sequence diagrams

i) Deposit - Refer Fig. 2.13.8.

ii) Withdraw - Refer Fig. 2.13.7.

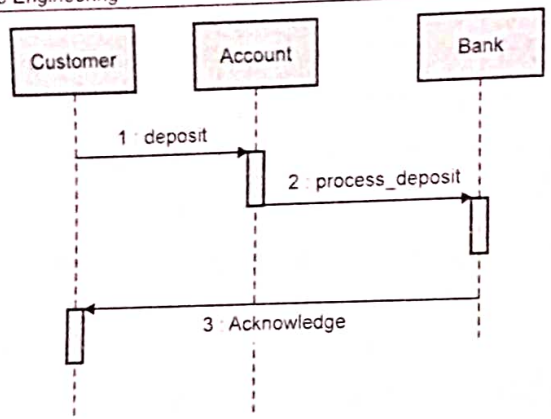


Fig. 2.13.8 Sequence diagram (Deposit)

2.13.3 Collaboration Diagram

- **Definition :** A collaboration diagram shows the objects and relationships involved in an interaction and the sequence of messages exchanged among the objects during the interaction.
- Collaboration diagram represent the structural organization of object.
- Both the sequence and collaboration diagrams are the **interaction diagram.**
- The sequence and collaboration are similar and can be transformed into each other.

2.13.3.1 Objects and Links

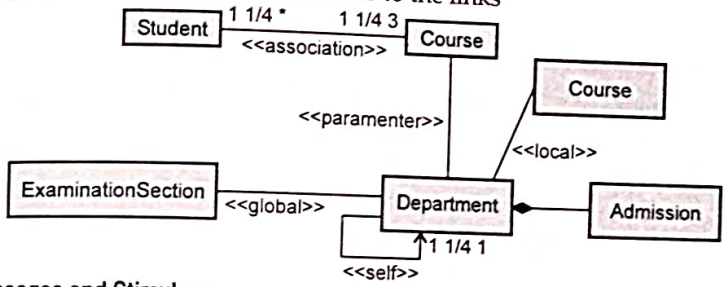
- For representing the structural organization of objects the collaboration diagram is used.
- The **communicating objects** are placed as vertices in graph. The **links** that connect these vertices are the arcs of this graph. These links have the role name to identify them.
- The links can be adorned using **messages** that are sent and received by the objects.
- This gives the **flow of control** in when the objects collaborate with each other.

A link specifies a path along which the message is sent to another object. The end of the link can be adorned by using following constraints : -

Adornment for end of links	Meaning
<<association>>	It specifies that corresponding object can be visible by the associating object.

<<global>>	It specifies that the corresponding object is visible in its enclosing scope.
<<local>>	It specifies that the corresponding object is visible in its as it has a local scope for invoking object.
<<self>>	It specifies that the object is visible due to its operations.
<<parameter>>	It specifies that the corresponding object is visible because it is parameter.

Following figure shows these adornments to the links

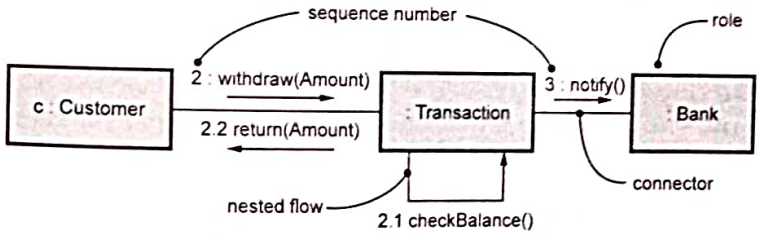


2.13.3.2 Messages and Stimulus

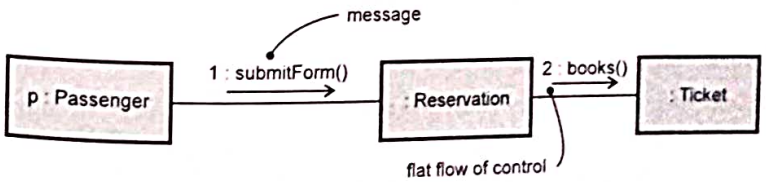
- There is a distinct flow of control within the system. For each control flow, the messages are ordered in some sequence. The sequence can be specified by following format

  Sequence_number : message

- Following is a communication diagram that shows the messages.



(a)



(b)

Fig. 2.13.9 Flat sequence

- There are two ways to specify the sequence of messages -
  - **Procedural sequence :** The procedural or nested flow of control is represented by a filled solid arrowhead. The nesting of some messages is represented by the sub-numbering.
  - **Flat sequence :** For representing the non procedural sequence, the flat sequence is used. It is represented by a stick arrowhead.
- The message sequence number is useful for distinguishing the flow of control.
- A stimulus represents a specific instance of sending the message, with particular arguments.
- **Types of messages :**
  - **Nested message :** The nested message represents a procedure call or other nested flow of control. The nested sequence is completed before the outer level sequence resumes. The nested message symbol is represented by a filled solid arrowhead.

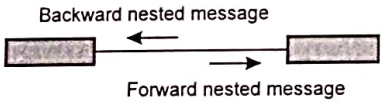


Fig. 2.13.10

  - **Flat message :** The flat message shows the progression to the next step in a sequence. The flat message symbol is represented by a stick arrowhead.

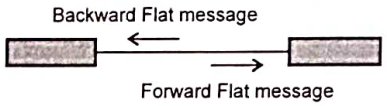


Fig. 2.13.11

  - **Asynchronous message :** The asynchronous message symbol shows an asynchronous message between two objects. The asynchronous message symbol is represented by a half stick arrowhead.

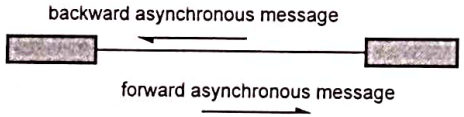


Fig. 2.13.12

2.13.3.3 Active Objects

Definition : An active object is an instance that owns a thread of control and can initiate control activity. For example, processes and tasks are active objects. Active objects can contain other symbols and links among them.

It can be represented as follows -

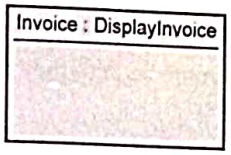


Fig. 2.13.13

The syntax of denoting the active object is

[name][:type]

where

name is the name of active object

type is a class or use case

2.13.3.4 Communication Diagram

There are two types of interaction diagrams - Sequence diagram and communication diagram. Both the diagrams are semantically equivalent. Hence one form of the interaction diagram can be converted to other without losing any information. But there are ways by which these two diagrams differ. For example in above given communication diagram, the two objects can be linked to other objects using {local} or {global} adornment but this is not possible in sequence diagram. Similarly, in sequence diagram at the end of the transaction you can destroy the object but this facility is not there in communication diagram.

Sequence diagram	Communication diagram
The sequence diagram models the lifeline of the objects. The lifeline denotes the existence of the object between its creation and destruction.	The communication diagram represents structural links among the objects during there interactions.
The particular object can be destroyed explicitly in sequence diagram.	There is no facility to destroy some object.
The linking of the objects with other object can not be represented as local or global.	The {local} or {global} scope of the objects with other objects can be represented in this diagram.

2.13.4 Strengths and Weakness of Sequence and Communication Diagrams

Different developers have different preferences for the interaction diagram. However the sequence diagram is preferred more because they posses greater notational power.

Strengths of sequence diagram

1. The call flow can be represented more systematically in sequence diagram than that of communication diagram.
2. The sequence diagram can be read from top to bottom and proved to be excellent for the documentation.
3. The reverse engineering can be easily carried out in sequence diagram.

Weakness of sequence diagram

1. Drawing sequence diagram is complicated.
2. It is more space consuming because if we wish to insert any new object it has to be inserted to the right.

Strength of communication diagram

1. The communication diagram can be easily drawn.
2. They are more space efficient because the boxes can be drawn vertically or horizontally and can be placed anywhere. Sketching of communication diagram is very simple.

Weakness of communication diagram

1. It is difficult to see the sequence of messages in communication diagram.
2. There are very few notational options in communication diagram.

2.13.5 Examples on Interaction Diagram

Example 2.13.2 *Draw a sequence diagram to show how a GUI interacts with other objects and also draw corresponding collaboration diagram.*

Solution :

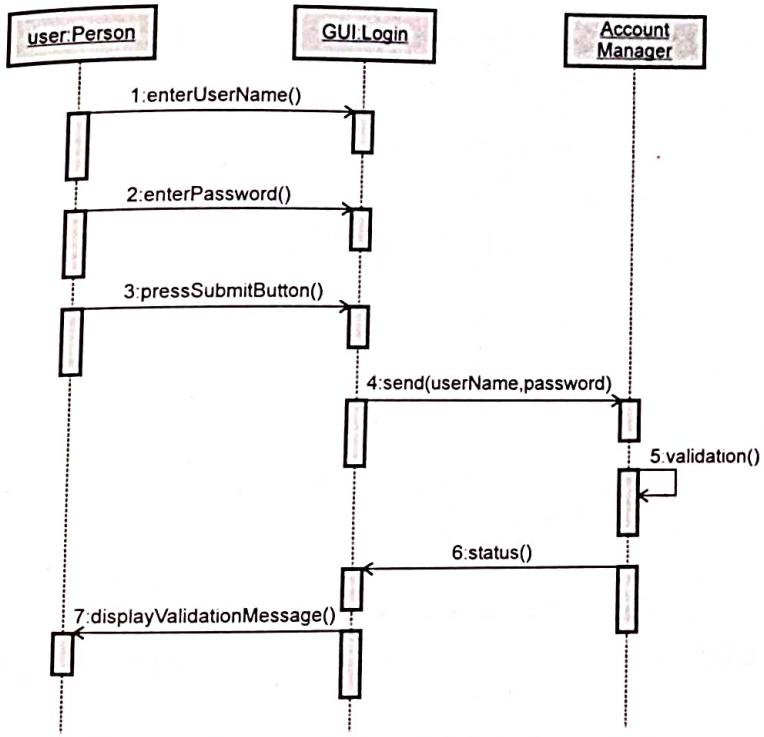


Fig. 2.13.14 (a) Sequence diagram for interaction with GUI

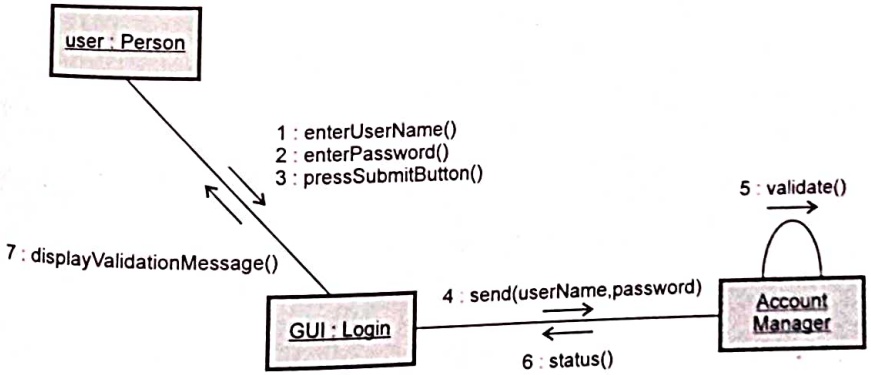


Fig. 2.13.14 (b) Communication diagram for interaction with GUI

Example 2.13.3 *Draw a sequence diagrams that specifies the flow of control involved in initiating a simple, two party phone call.*

Solution :

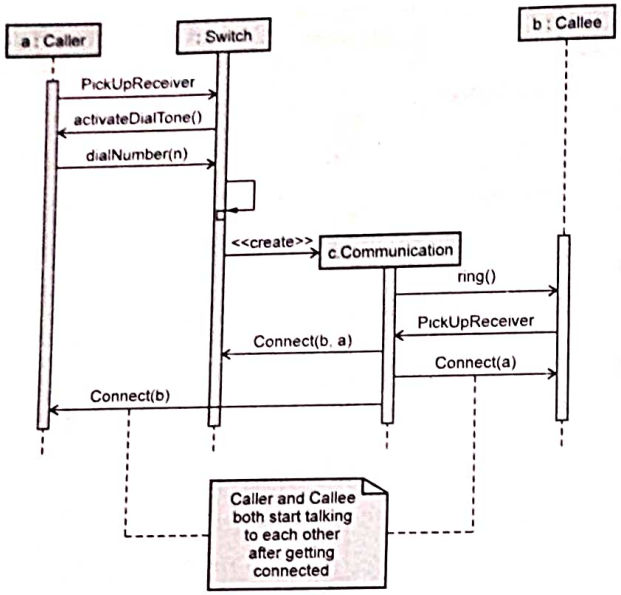


Fig. 2.13.15 Sequence diagram for two party phone call

Example 2.13.4 *Draw a collaboration diagram that specifies the flow of control involved in registering a new student at a school.*

Solution :

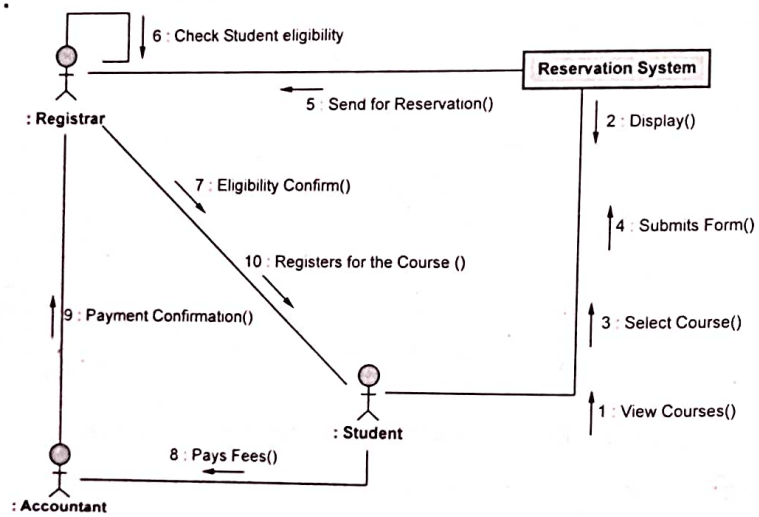


Fig. 2.13.16

Example 2.13.5 *Draw sequence diagram for login operation in ATM model.*

Solution : The login operation in ATM model can be performed by entering the valid ATM card and PIN. The sequence diagram for it is given below -

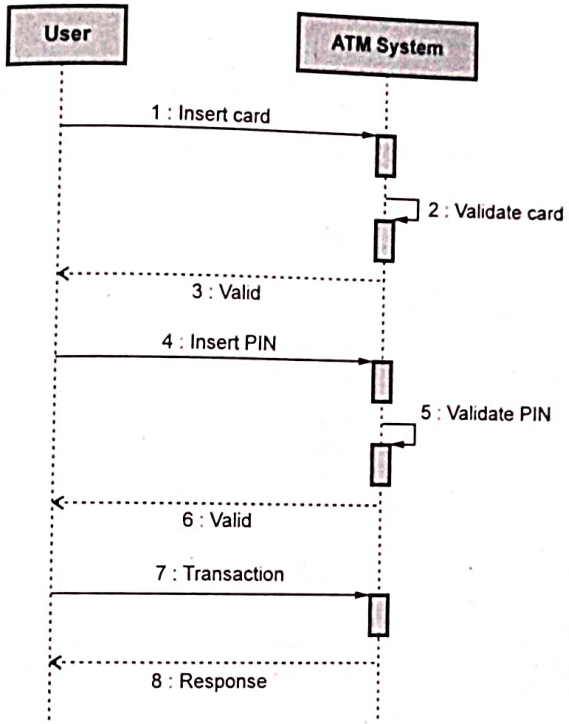


Fig. 2.13.17

Example 2.13.6 *Explain the two interaction diagrams for "issue of a book" and "renewal of a book" operations in a library.*

Solution : i) Sequence diagram for issue of a book

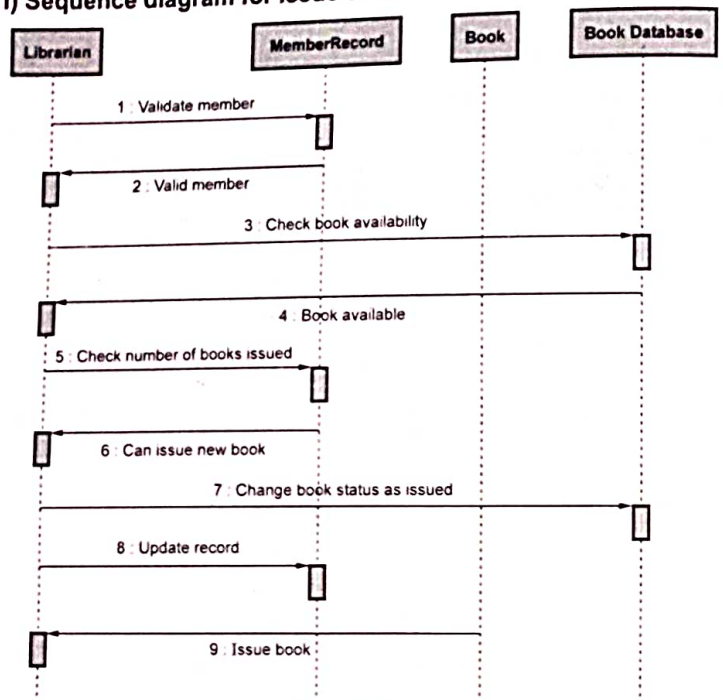


Fig. 2.13.18

ii) Renewal of a book

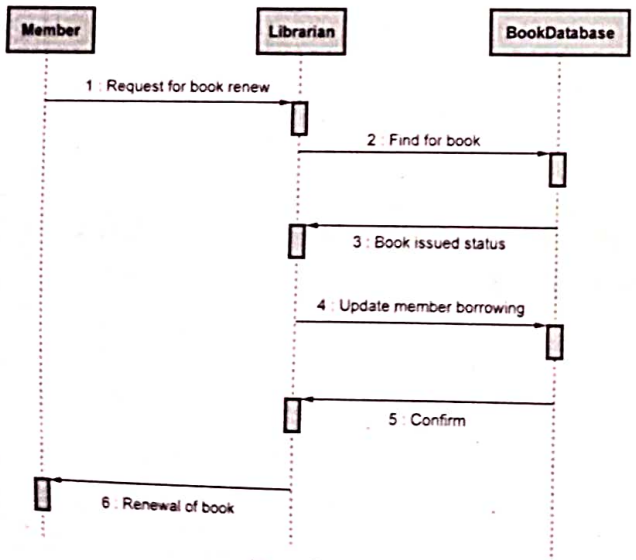


Fig. 2.13.19

Example 2.13.7 *Draw sequence and collaboration diagram for online course reservation system.*

Solution : i) Sequence diagram

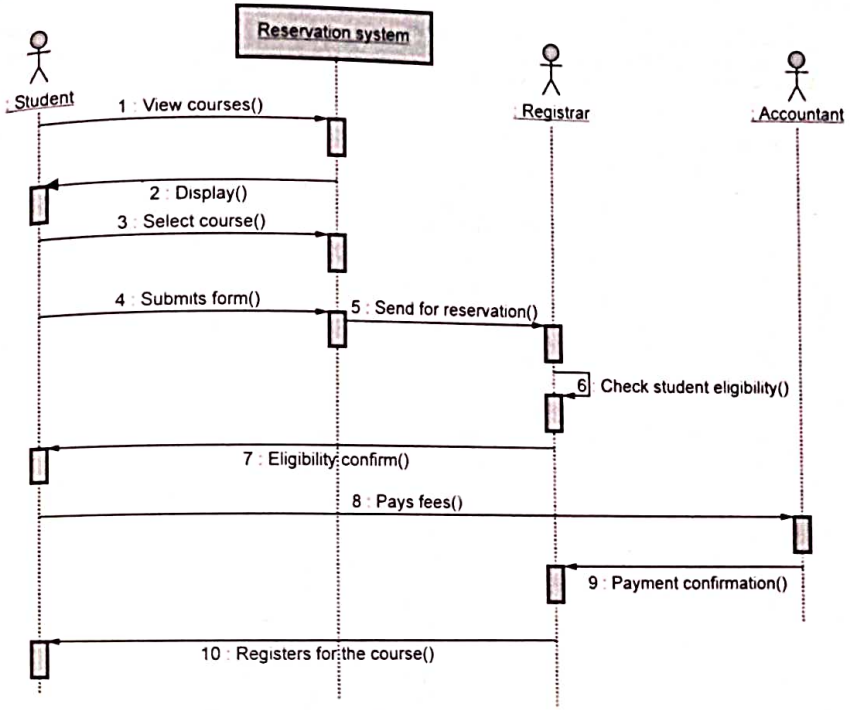


Fig. 2.13.20

Collaboration diagram

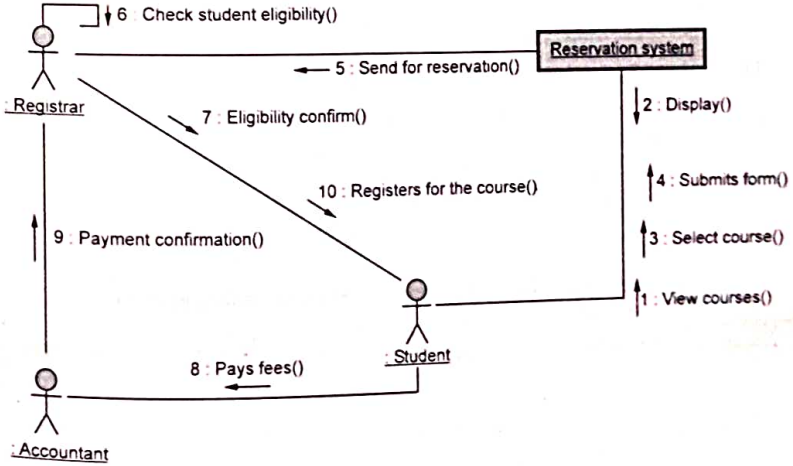


Fig. 2.13.21

Example 2.13.8 *Draw a sequence diagram for online shopping system.*

Solution :

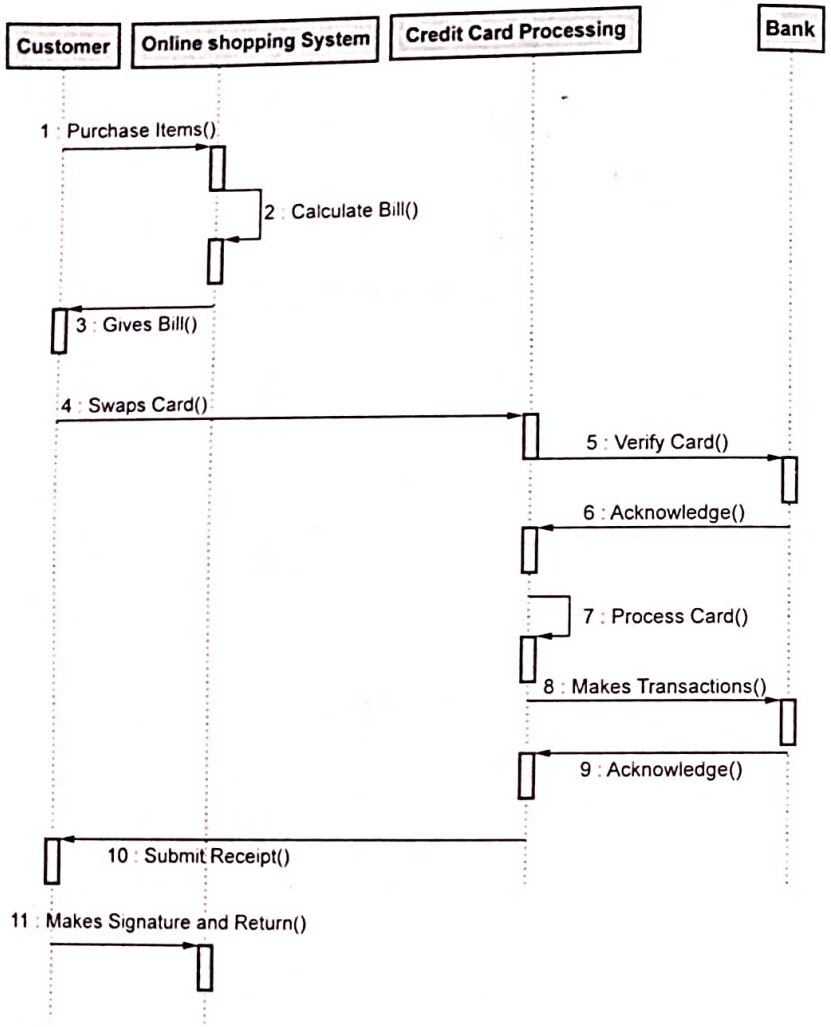


Fig. 2.13.22 Sequence diagram for online shopping system

Example 2.13.9 *Explain about interaction diagram notation for inventory management system.*

Solution :

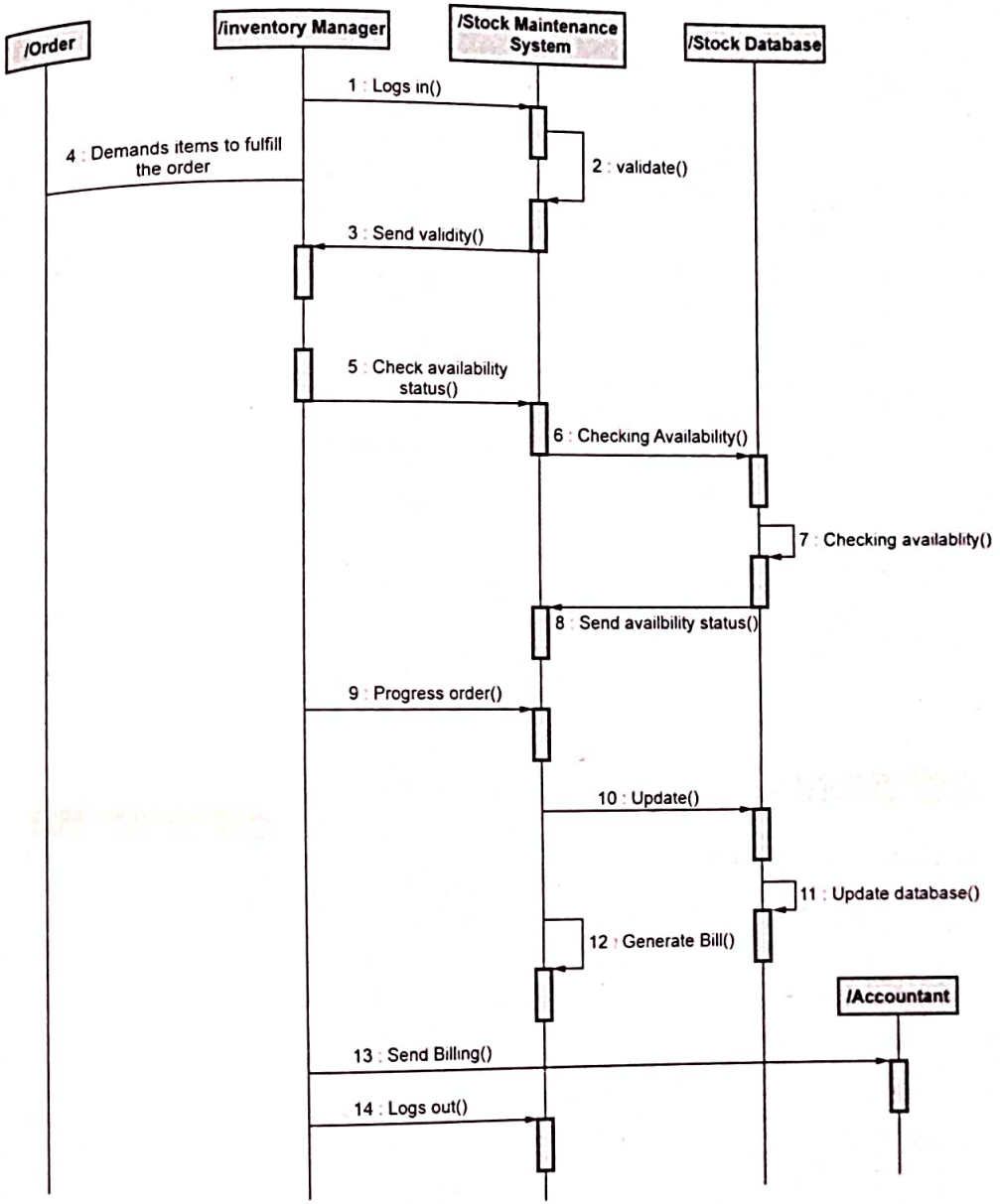


Fig. 2.13.23

Example 2.13.10 *The use case 'Login' accepts userid and password from the user. The system validates them and displays a message 'Log in is successful'. Draw a sequence diagram for this.*

Solution :

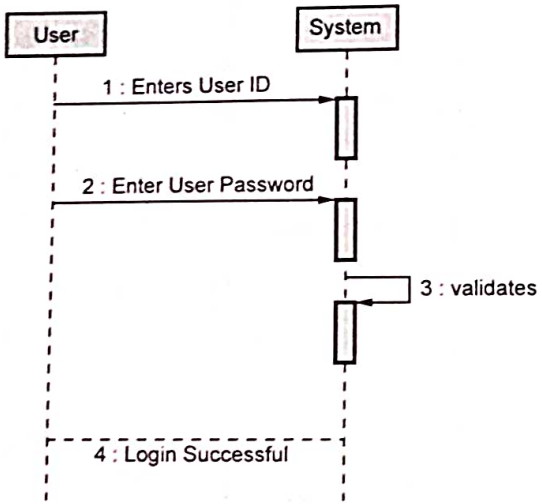


Fig. 2.13.24

Example 2.13.11 *Explain about Interaction diagram notation for inventory management system.* **AU : Dec.-15, Marks 16**

Solution : See Fig. 2.13.25 on next page

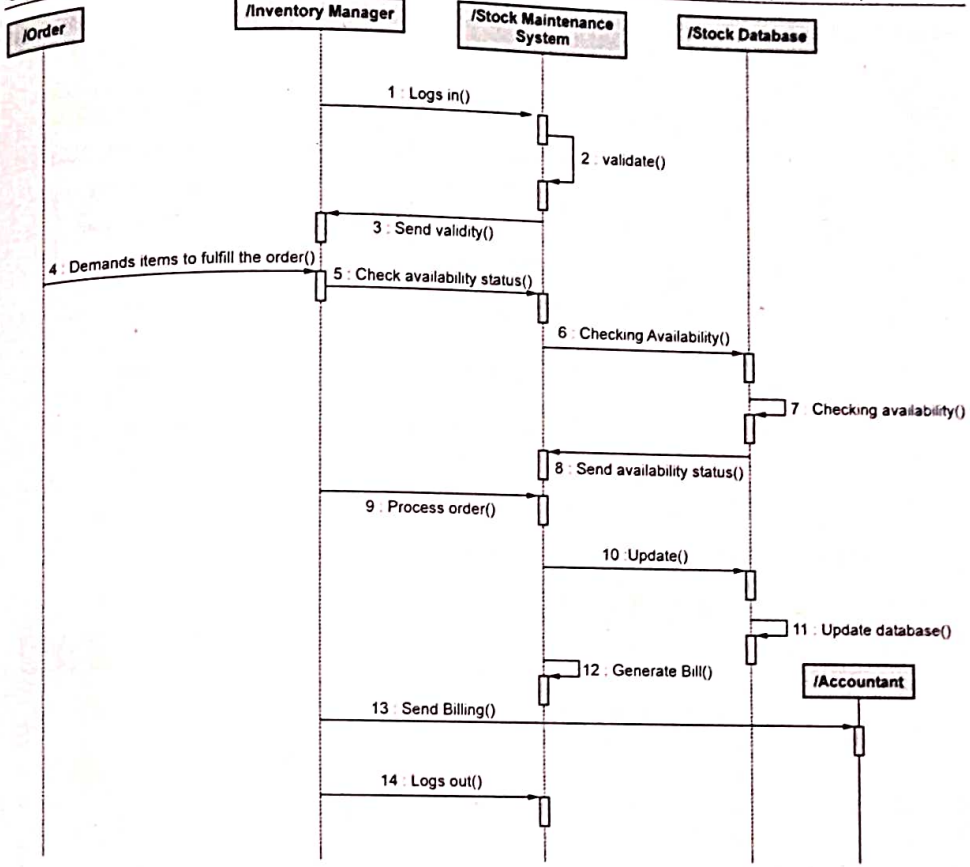


Fig. 2.13.25

Example 2.13.12 *Apply interactive modeling for a payroll system in UML.*

AU : Dec.-16, Marks 16

Solution : The interaction modeling can be done by designing the sequence diagram for the given problem statement. The sequence diagram for payroll system is as shown below

Sr. No.	Sequence diagram	Collaboration diagram
1.	The sequence diagram models the **lifeline of the objects**. The lifeline denotes the existence of the object between its creation and destruction	The collaboration diagram represents **structural links** among the objects during there interactions.
2.	The particular object can be destroyed explicitly in sequence diagram.	There is no facility to destroy some object.

3.	The linking of the objects with other object can not be represented as local or global.	The {local} or {global} scope of the objects with other objects can be represented in this diagram.
4.	Designing sequence diagram is very difficult.	It is difficult to see the sequence of message in collaboration diagram.

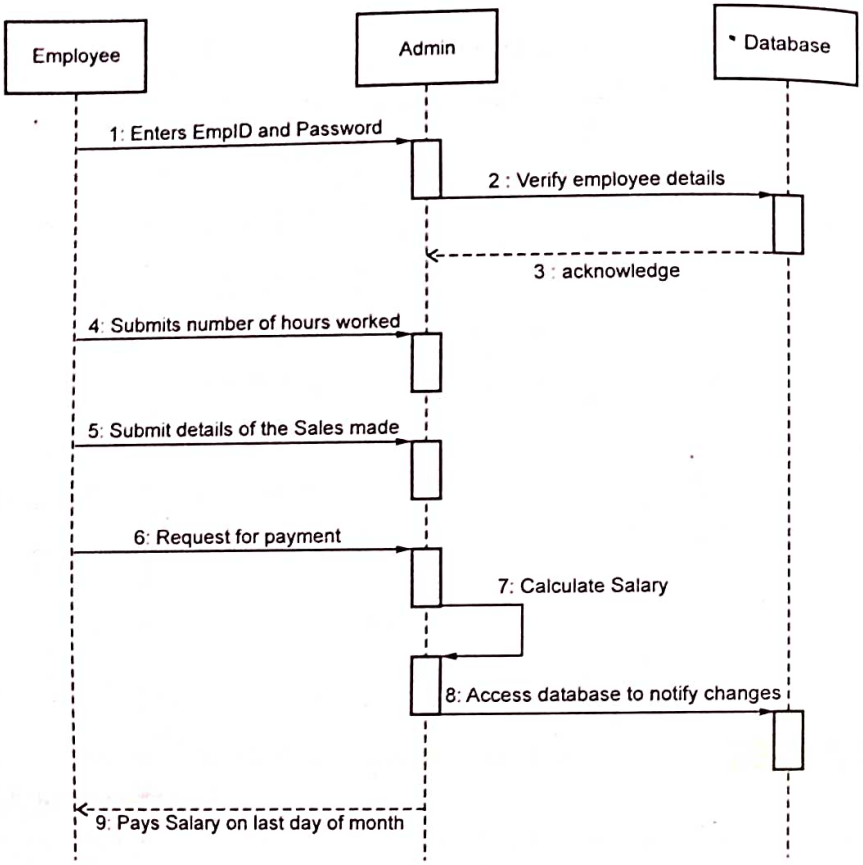


Fig. 2.13.26

2.13.6 When to use Sequence Diagram ?

The sequence diagram represents following things -

- One particular scenario of a use case
- The events that are generated by the external actors
- Sequence of those events
- Events among the interacting objects of the system.

2.13.7 When to use Communication Diagram ?

The interaction diagrams such as sequence diagram or communication diagrams are used to represent the dynamic aspect of the system. Following are the two ways by which the communication diagram is used -

1. To model flows of control by time ordering

- The communication diagram is used to model the flow of control by time ordering.
- The sequence of messages that take part in the interaction over the time is represented in the communication diagram.
- The iterations and branching are shown more effectively in communication diagram.

2. To model flows of control by organization

- The communication diagram us used to model flow of control by organization of objects.
- This diagram emphasizes on the structural relationship. The interaction among various instances (objects) is shown by the messages.

Review Questions

1. *Discuss the relationship between sequence diagram and class diagram.* **AU : Dec.-18, Marks 13**
2. *Illustrate with an example, the relationship between sequence diagram and use cases.* **AU : May-11,14, Dec.-13,14, Marks 16, May-18,19, Marks 13**
3. *What are system sequence diagrams ? What is the relationship between SSDs and Use cases ? Explain with an example.* **AU : Dec.-16, Marks 16**
4. *Mention the strengths and weakness of sequence diagram vs. communication diagram.*
5. *What is the use of system sequence diagram ?* **AU : May-11, Marks 2**

2.14 Activity Diagrams

AU : Dec.-13,18, Marks 16

- An activity diagram, is just similar to the flowchart in which flow of control from activity to activity is shown. The difference between traditional flow chart and activity diagram is that - in an activity diagram, the concurrency as well as branches of control is shown.
- Activity diagram focuses on dynamic aspect of the system. An activity is an ongoing structured execution of behavior.

2.14.1 Activity and Actions

Actions

- The executable and atomic computations are called as actions. For example - evaluation of some expression, sending of signal to some object, creation or destruction of some object - all these represent the actions.
- There are two ways of representing an action - 1) Simple action and 2) expression.
- The actions are represented by rounded box.

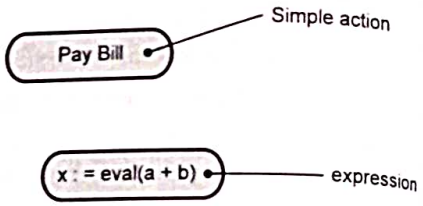


Fig. 2.14.1 Actions

Activity node

- Activity node is a nested grouping of actions or other nested activity node.
- Activity nodes are created for those actions that take time to complete.
- Actions is an activity node that cannot be further decomposed.
- It is represented as

Fig. 2.14.2 Activity nodes

2.14.2 Initial and Final Activity

- **Initial state :** In activity diagram, the initial state is represented by a solid circle. From this circle the transition to the first activity starts. Thus the initial state represents that all the activities start their execution from this point.
- **Termination :** In activity diagram, the termination is represented by **bull's eye** i.e. a solid circle surrounded by a hollow circle. This state represents the termination of all the activities at this point. Fig. 2.14.3 represents these states.

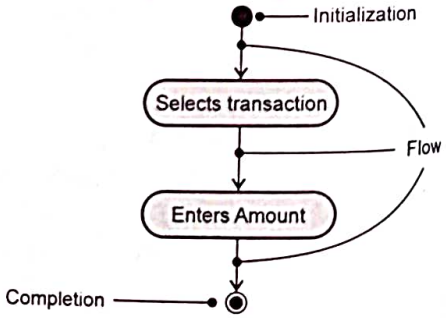


Fig. 2.14.3 Initiation and termination

2.14.3 Activity Edge

- In order to show the flow through an activity the actions are linked up using **activity class.**
- Activity edge can be **control edge** or **data flow edge** (object flow edge).
- Thus we can show activity edge as a line with an arrow pointing to the next action. For example :

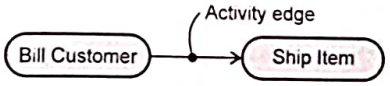


Fig. 2.14.4 Activity edge

2.14.3.1 Object Flows

When activity edge is represented for data only. For example,

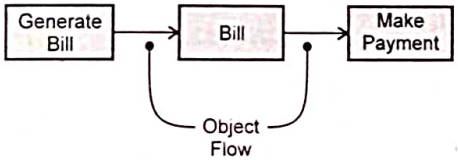


Fig. 2.14.5

2.14.4 Decision and Merge Points

A decision node or decision point is a control node which has one input edge and multiple outgoing edges.

When data arrives at the decision point, a single outgoing edge is selected from multiple outgoing edges and data is sent over that outgoing edge. The outgoing edge is selected based on the evaluation of guard condition. Guard condition is a boolean expression that tests some value visible to the activity.

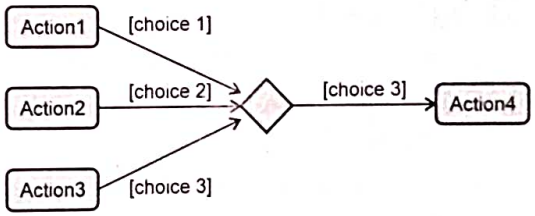


Fig. 2.14.6 Merge point

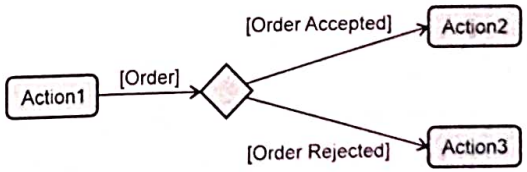


Fig. 2.14.7 Decision point

The merge node is opposite of decision node. It has multiple input flows and single output flow. The merge point simply takes any token along some incoming input flow and makes it available on the outgoing edge.

The notation for decision and merge point is diamond shaped.

Example

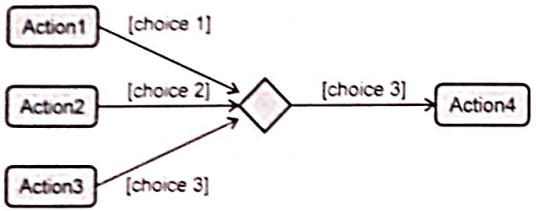


Fig. 2.14.8 Merge point

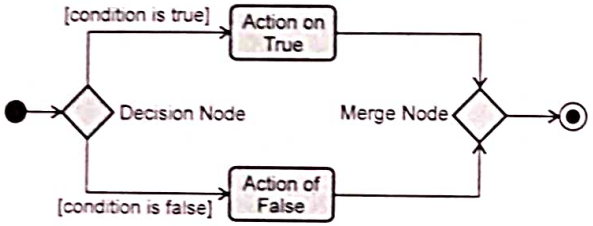


Fig. 2.14.9 Decision and merge point together

2.14.5 Fork and Join

- There are two types of activities - sequential activities and concurrent activities.
- The activities that can be performed one after the other are called sequential activities.
- The computer system can perform more than one activities at a time. The activities that can be performed at the same time are called concurrent activities.
- The concurrent activities can split in several activities or can join and merge in some action.
- In activity diagram the concurrent activities are represented with the help of **forking and joining**.
- The fork and join in activity diagram is represented by **synchronization bars**.
- The synchronization bars are represented by thick horizontal or vertical bars.
- The **fork** represents the separation of two control flows whereas **join** represents joining of two control flows.

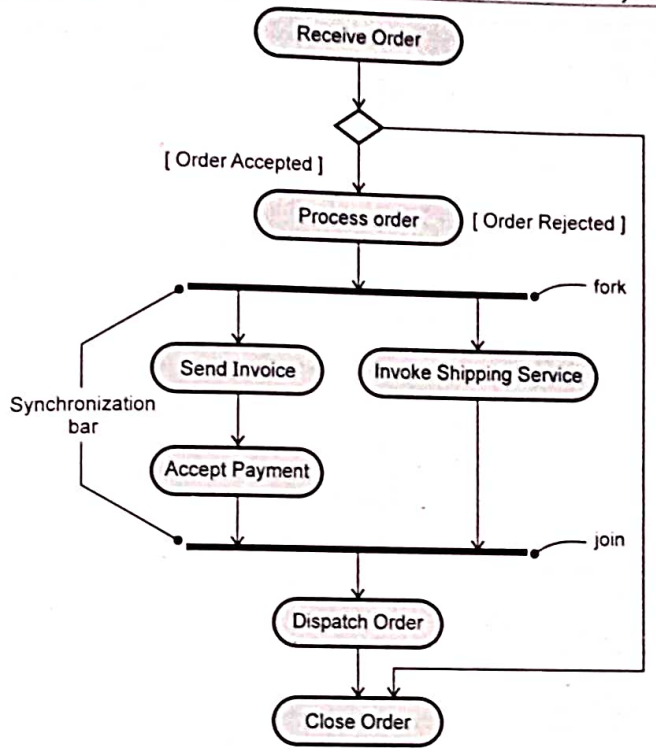


Fig. 2.14.10 Forking and joining

2.14.6 Activity Partitions or Swimlane

- Many times the activity diagram needs to partition in groups to represent the flow of business processes.
- Each group represents the particular business flow belonging to some category. Such a group is called **swimlane.**
- Graphically the swimlane is denoted by a vertical solid line.

2.14.7 Object Flow

- The objects may be involved in the activity diagram.
- These objects are associated with the control flow of the activity diagram. This is called **object flow.**
- The state of the object is specified for each of these objects.
- Following is a simple activity diagram in which the object flow is represented for the objects of the classes **Order** and **Invoice**.

(See Fig. 2.14.11 on page 2 - 120)

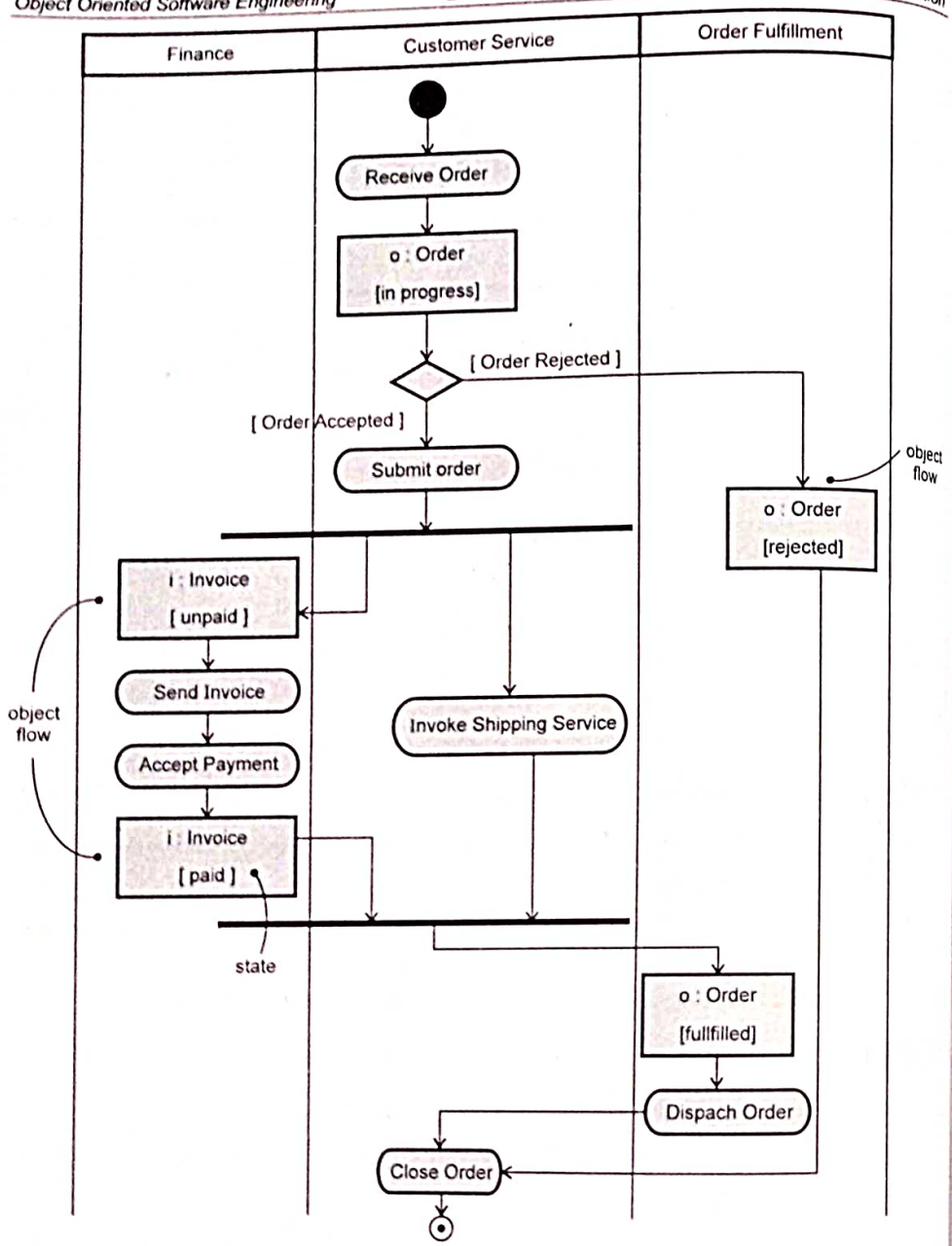


Fig. 2.14.11 Object flow

2.14.8 When to use Activity Diagrams?

- Activity diagrams are used to model the **dynamic aspects** of a system. That means the activity involved in classes, interface, components and nodes can be represented in the activity diagram.

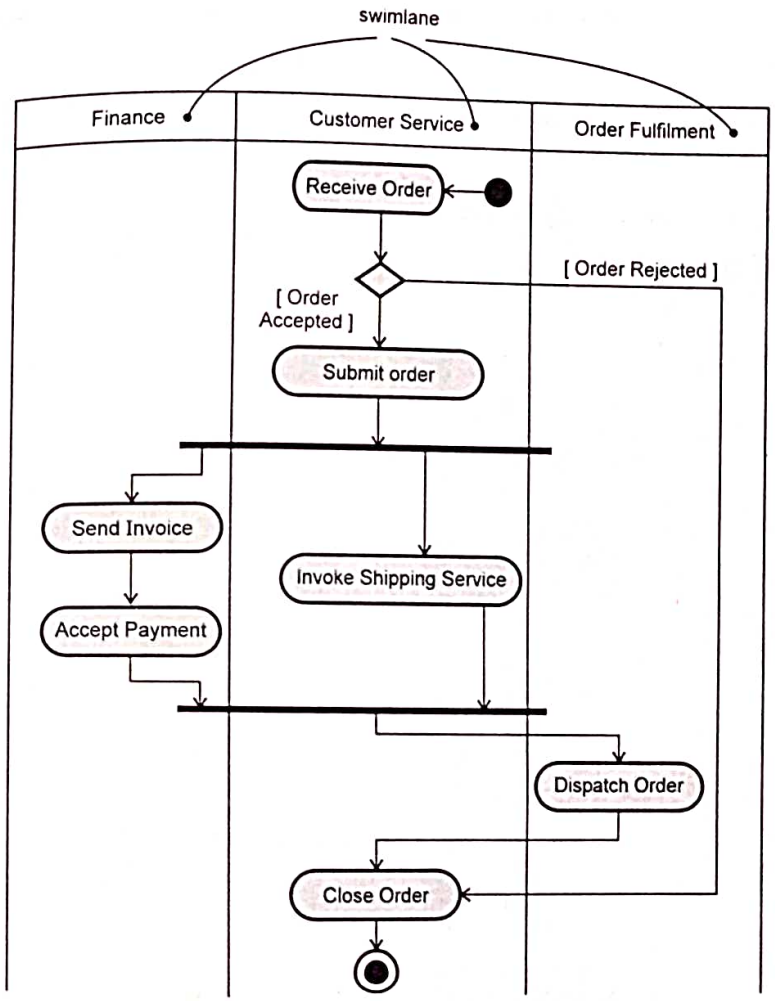


Fig. 2.14.12 Swimlane with an activity diagram

- The activity diagram can be used in two ways -

1. To model the workflow

- The actors collaborate with the system and then various activities are performed.

- These activities depict the object flow.
- The activity diagrams can be used to model this work flow.

2. To model the operations

- In this case the activity diagram is used as a flowchart to model some computations.
- These activity diagrams include branch, fork and join. Such activity diagrams are normally useful for design of the operations having parameters and local objects.

2.14.9 Examples on Activity Diagram

Example 2.14.1 *Draw activity diagram for issue of book in library information system.*

Solution :

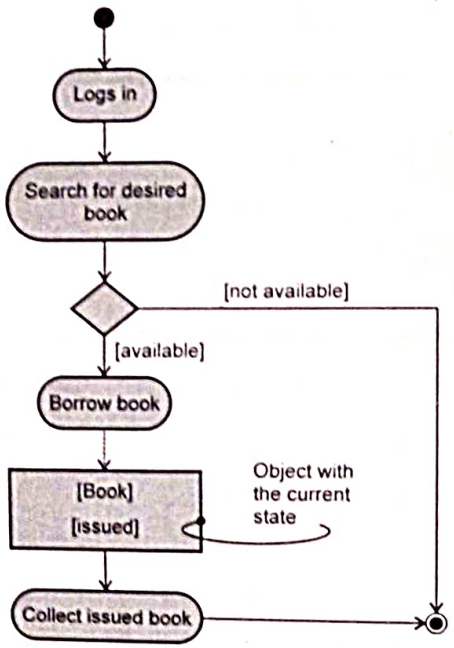


Fig. 2.14.13

Example 2.14.2 *Draw swimlane diagram for exam registration systems.*

Solution :

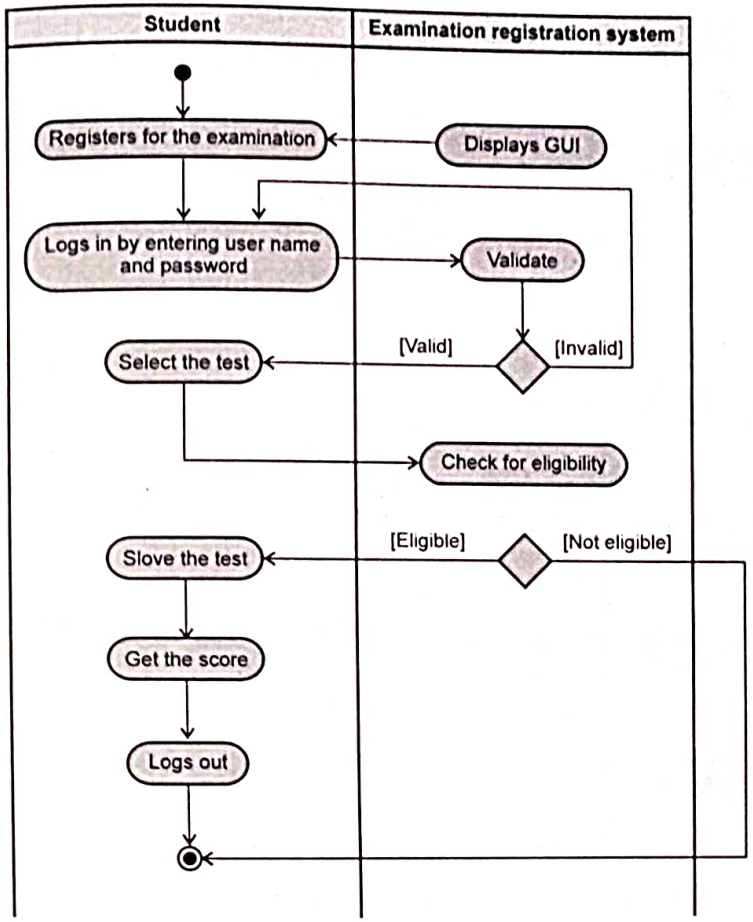


Fig. 2.14.14

Example 2.14.3 *Draw activity/swimlane diagram for stock maintenance systems.*

Solution :

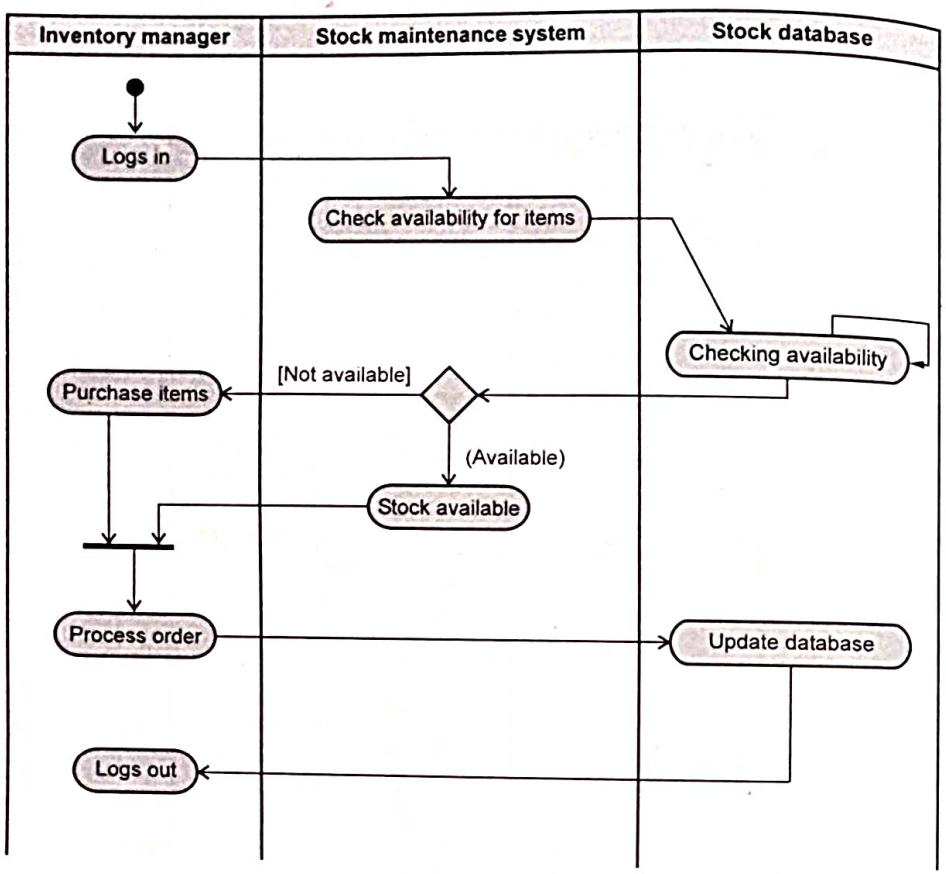


Fig. 2.14.15

Example 2.14.4 *Draw activity/swimlane diagram for credit card processing system.*

Solution :

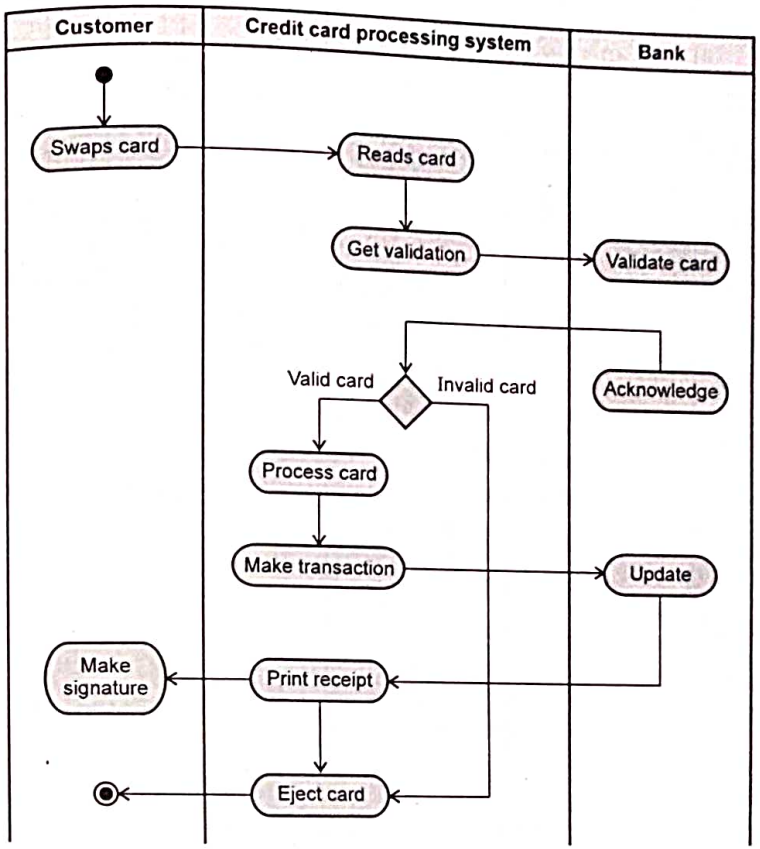


Fig. 2.14.16

Example 2.14.5 *A customer decides to upgrade her PC and purchase a DVD player. She begins by calling the sales department of the PC vendor and they tell her to talk to customer support. She then calls customer support and they put her on hold while talking to engineering. Finally the customer support tells the customer about several supported DVD options. The customer chooses a DVD and it is shipped by the mail department. The customer receives the DVD, installs it satisfactorily and then mails her payment to accounting.*

Construct an activity diagram for this process. Use swimlanes to show the various interactions.

Solution :

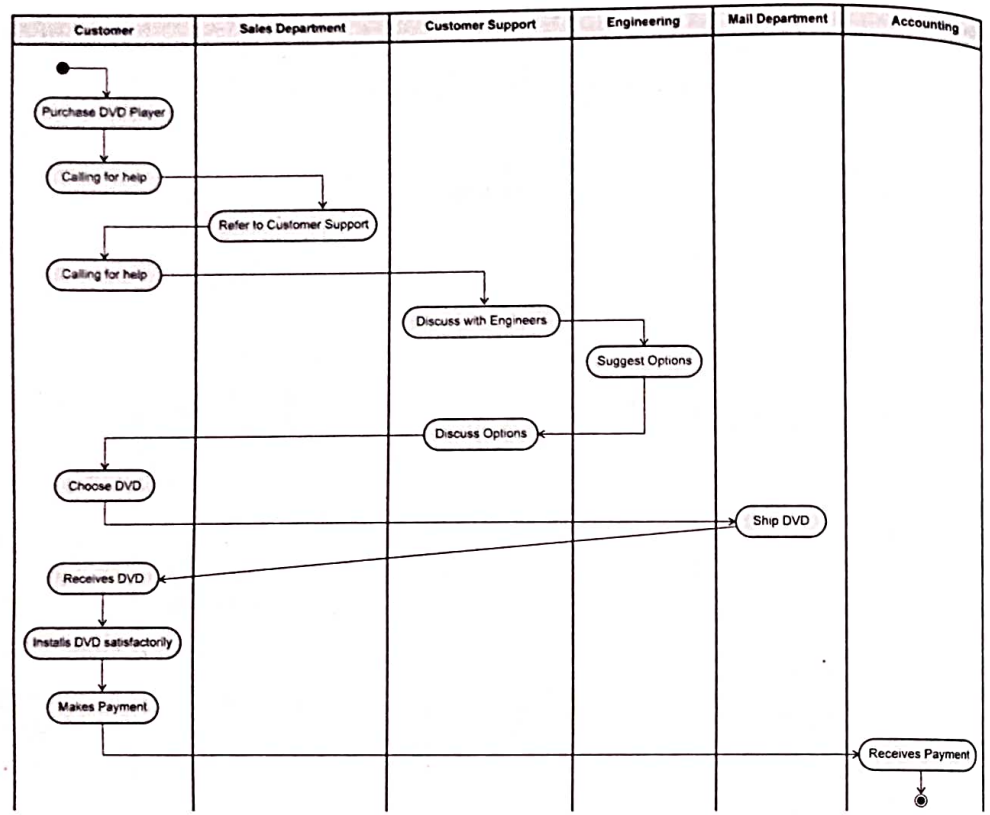


Fig. 2.14.17 Swimlane diagram

Example 2.14.6 *Prepare an activity diagram for awarding marks to regular students. If the student has attended 80 % classes, he is awarded minimum 5 marks. If the student has attended more than 80 % classes, he is awarded minimum 10 marks. The students who have completed assignments are given 10 marks. Those who have completed 50 % are given 5 marks and rests are given 0 marks.*

Solution :

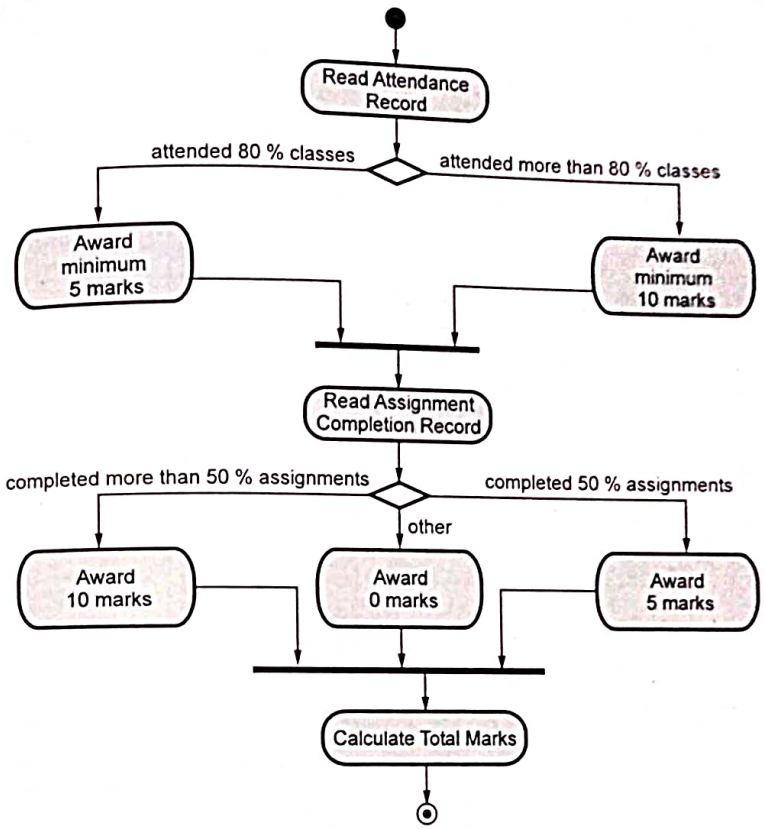


Fig. 2.14.18

Example 2.14.7 *A draw an activity diagram for a Passport Management System to get a new passport, an applicant has to apply on-line, get the appointment. He has to submit the documents in passport office on the date of appointment. In case of insufficient or incorrect documents, the applicant's has to reapply and get new appointment. After submission of documents, applicant's verification is done by the police. On successful verification, passport is issued to the applicant. If verification is unsucessful, applicant has to reapply for passport.*

Solution :

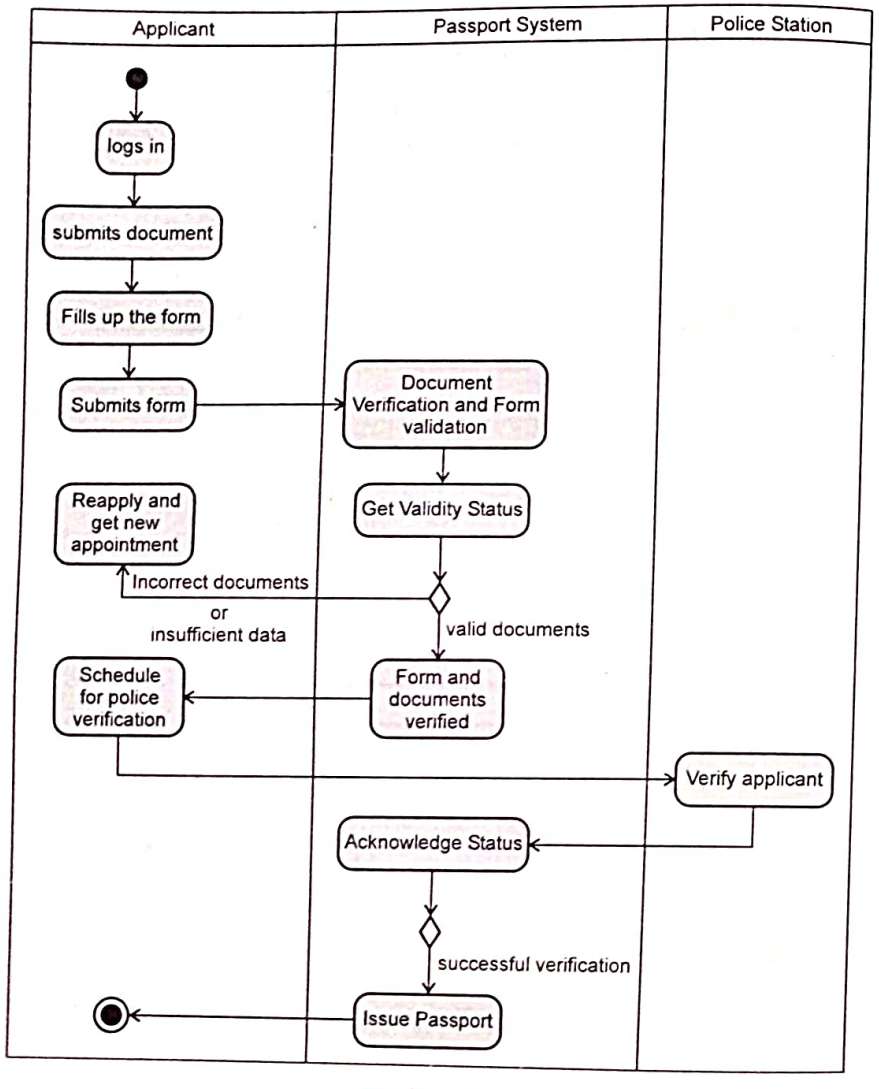


Fig. 2.14.19

Example 2.14.8 *Insurance system provides vehicle insurance to the owners. Initially the customer fill the form which has vehicle details and owner's details. This information is submitted to the agent. The agent sends the information to various insurance companies. The companies quote the insurance. The agent selects the best policy and gives it to the owner. Draw an activity diagram with swimlane and other applicable notations.*

Solution :

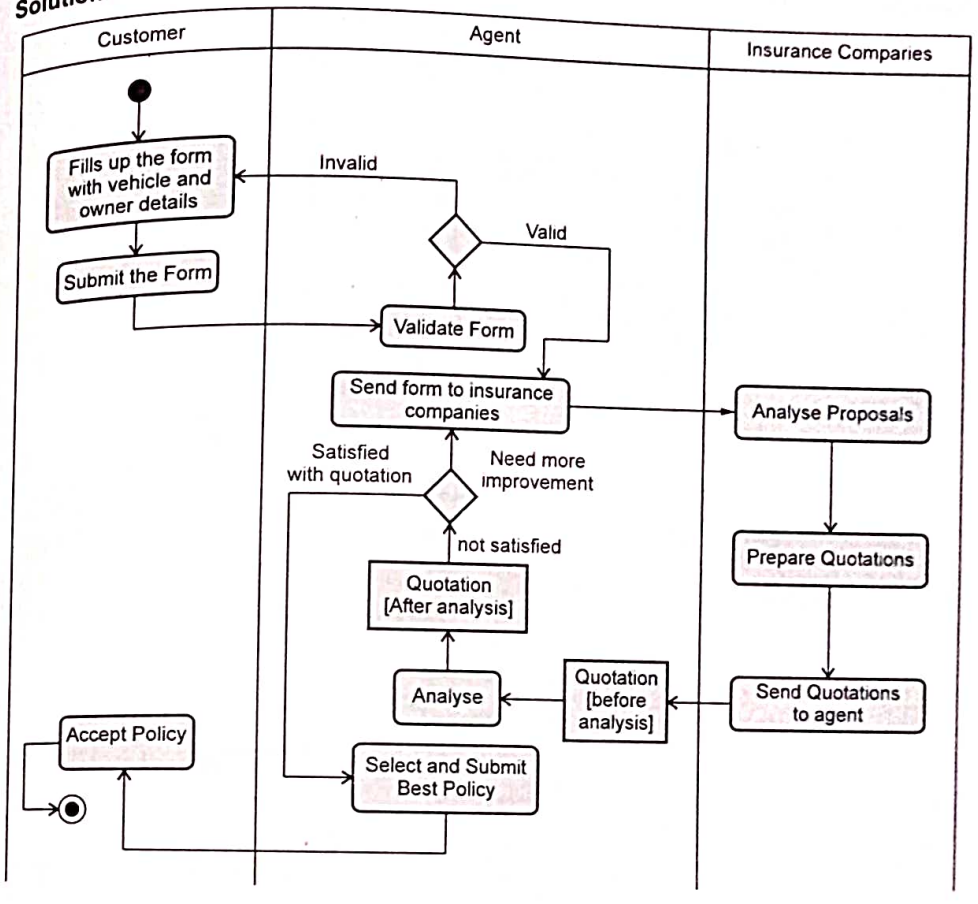


Fig. 2.14.20

Example 2.14.9 *A coffee vending machine dispenses coffee to customers. Customers order coffee by selecting a recipe from a set of recipes. Customers pay for the coffee using coins. Change is given back, if any, to the customers. The 'Service Assistant' loads ingredients (coffee powder, milk, sugar, water, chocolate) into the coffee machine. The 'Service Assistant' adds a recipe by indicating the name of the coffee, the units of coffee powder, milk, sugar, water and chocolate to be added as well as the cost of the coffee. The Service Assistant can also edit and delete a recipe. i) Develop the use case diagram for the specification above. (Marks 6) ii) For any two scenarios draw an activity diagram and sequence diagram. (5+5 Marks)* **AU : Dec.-13, Marks 16**

Solution : i) The use case diagram is as given below -

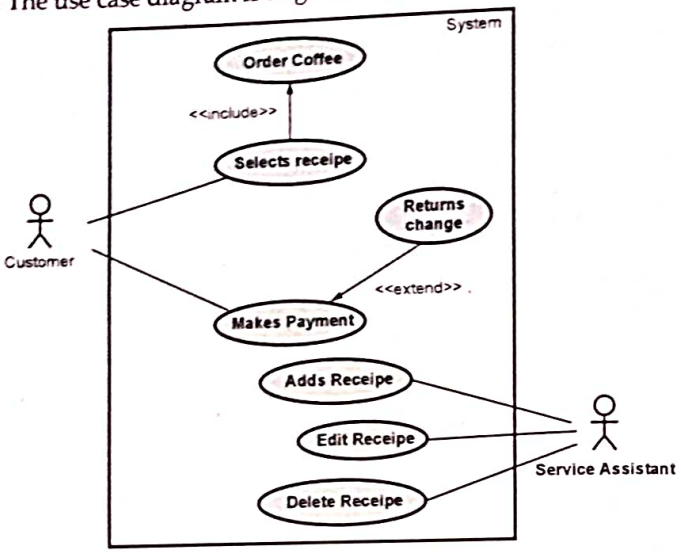


Fig. 2.14.21

ii) Activity diagram

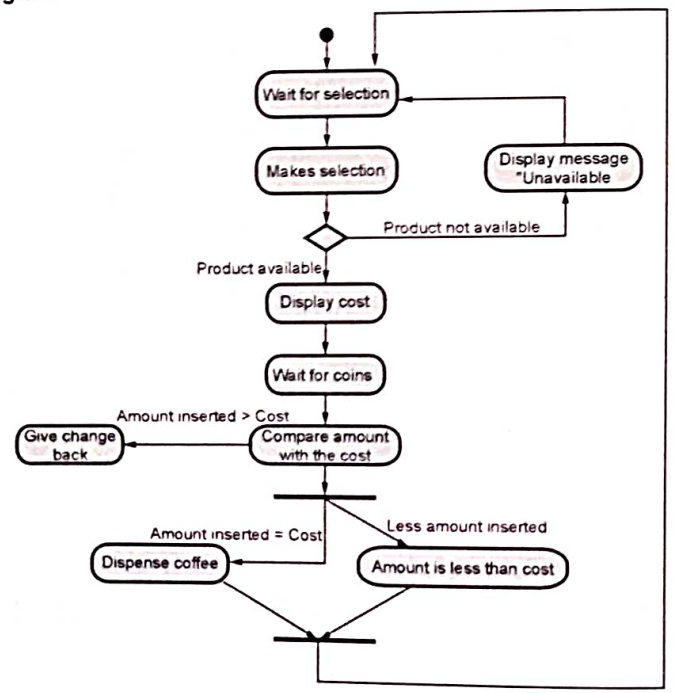


Fig. 2.14.22 Activity diagram for customer orders for the coffee

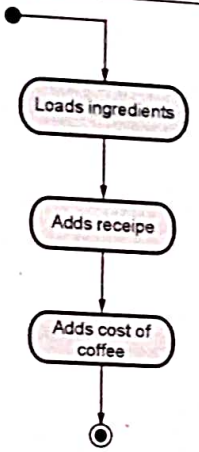


Fig. 2.14.23 Activity diagram for service assistant for adding the receipe

Sequence diagram :

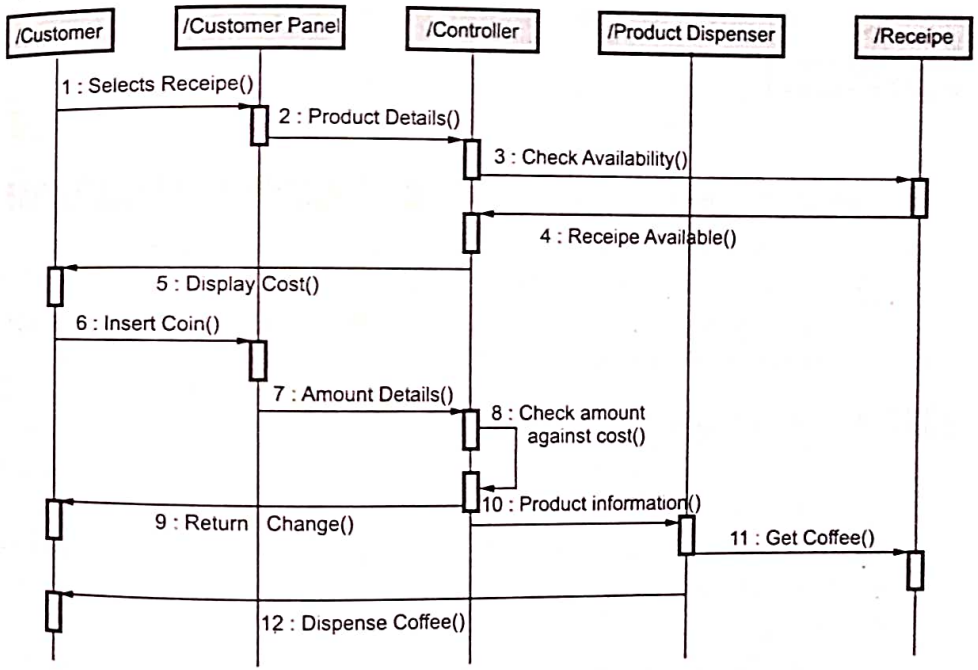


Fig. 2.14.24 Customer getting coffee from vending machine

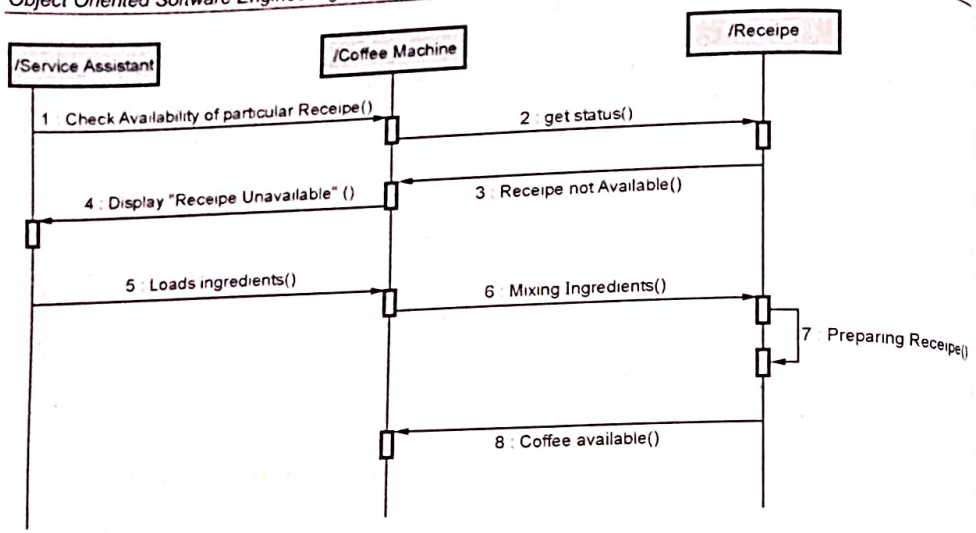


Fig. 2.14.25 Service assistant loads the ingredient and prepare coffee

Review Question

1. *With an example, Explain the need for activity diagram.* **AU : Dec.-18, Marks 13**

2.15 State Chart Diagrams

AU : Dec.-11,13,14, May-11,14,19, Marks 16

Purpose : UML **State machine diagram** represents the events and states of object and the behavior of object in reaction to an event.

The state chart diagram shows the life cycle of the object. Sometimes it is not possible to show each and every event in the state diagram.

2.15.1 Events, States and Transitions

- **Event :** An event is significant occurrence of something.

  For example : Customer inserts ATM card.
- **States :** A state is a condition of object at a particular moment of time. This time is mostly between the events.

  For example :

1. Waiting for user to enter PIN 2. Authenticating the User 3. Dispensing Cash

Note that these states represent that object is waiting for something or doing something.

- **Transitions :** The relationship between the two states is indicated using the transition. When some event occurs the object moves or transits from one state to another. This is represented by the transition.

For example : When the system operator "performs startup"(an event occurs) the transition from **Idle state** to **Active state** occurs.

Example :

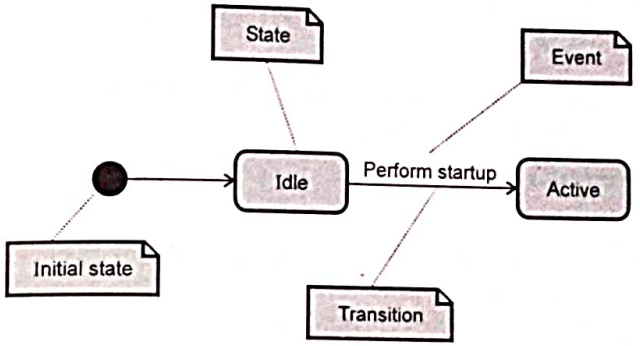


Fig. 2.15.1 State machine diagram for ATM system

2.15.2 Applying State Machine Diagram

If the object responds to the same way to an event then that object is considered as **state-independent** with respect to that event. For example - If an object receives the message "Customer finished the transaction" or "Customer pressed the Cancel button" the corresponding method will be **Ejecting Card.**

If objects respond differently to the events depending upon their states or mode then such objects are considered as **state-dependant** objects.

For example - For an "Insert Card" event the state will be "Reading Card" whereas "Enter PIN" event the state will be "Validating PIN"

Guideline about the state dependant and independent objects

1. The state machines containing state dependant objects with complex behaviour is considered better than the state independent objects.
2. The business information systems have few complex state-dependant classes. On the other hand the process control, device control and protocol control systems have many state dependant objects.

The state machine diagrams can be created for following two good reasons -

1. In order to model the complex reactive objects in response to events.
2. To model legal sequence of operations or language specifications.

A list of common objects used in state model diagram is -

1. Complex reactive objects

- The **physical devices** controlled by the objects can be represented as the state dependant objects. For example CardReader, Microwave Oven, Washing Machine.
- The **transaction** or **business objects** such as Order, Invoice Management, Purchase can be state dependant objects.
- The **Role mutator** are the objects that change their role. For example **Student**.

2. Protocols

- The communication protocols such as **TCP** or **UDP** can be understood using the State machine model.

3. Legal sequences

- **User Interface Controller** or **session** such as web applications in which modified web pages can be displayed to the user based on the current session.
- **Navigation** can be represented using the state machine model. The navigation represents the transition of one web page to another or transition of windows.
- For representing **Individual UI Window event handling** the state transition diagram is an effective tool. The sequence of occurring events can be represented using the state dependant object.
- Use case operations can be represented using the state machine model. For example - Withdrawal of money, Validating user.

2.15.3 More Notations

During the transactions the event occurs and these events cause the change in state. These transition may represent the call to the method of some specific class. A transition may be a condition guard which represents the Boolean test. When the condition results true then only the further state change occurs. For example in the following Fig. 2.15.2 the transition between the *Reading Card* and *Requesting PIN* states is shown. On reading card, it is necessary to validate that card. If the inserted card is valid then only the request for the PIN is made. When the process of Reading card gets completed then the indicator is glowing.

The *Glow indicator* denotes the transition action and the *Valid Card* denotes **guard condition.**

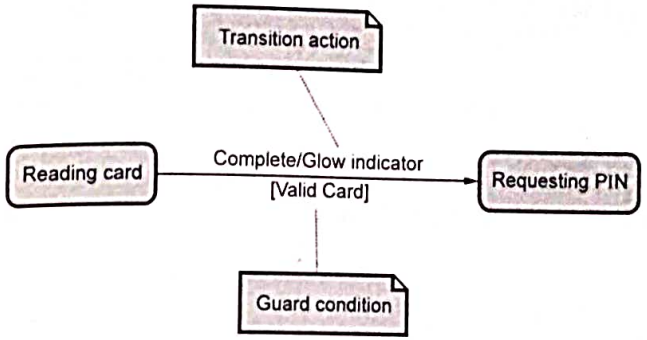


Fig. 2.15.2

Nested states

The nested states are the states that occur within the one super state. For example in the following state machine diagram, the **Active** state is a super state in which the sub states such as **Display, Reading Card Validating PIN** and so on are present. When the Perform shutdown event occurs then the system is restored to the **Idle** state from the **Active state.**

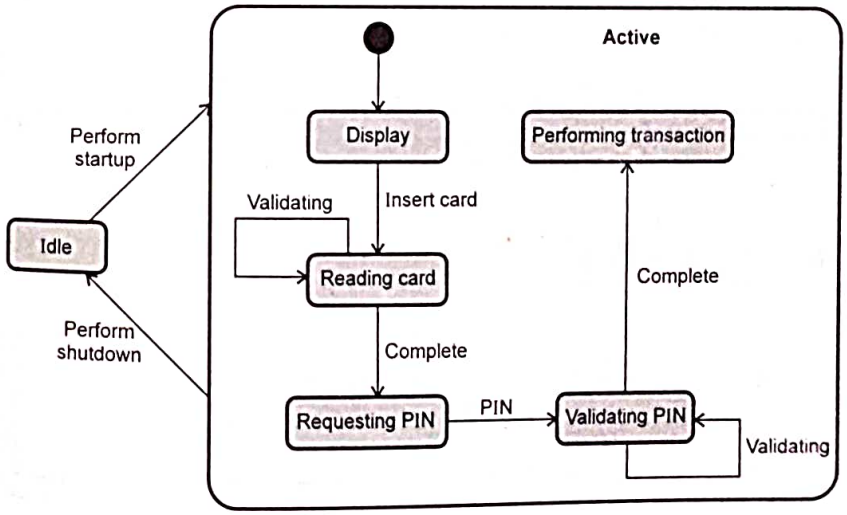


Fig. 2.15.3

Example 2.15.1 *Following is a simple state machine diagram representing one session with ATM System*

Solution :

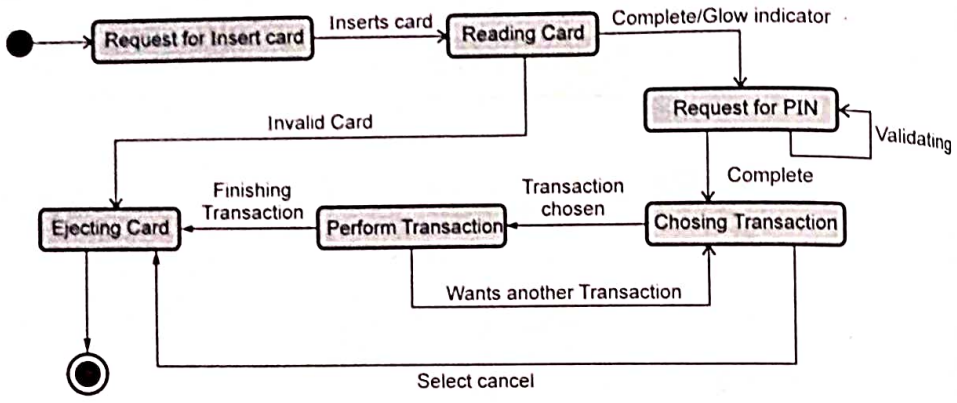


Fig. 2.15.4 State machine model

Example 2.15.2 *Draw state diagram for the control of a telephone answering machine. The machine detects an incoming call on the first ring and answers the call with a prerecorded announcement. When the announcement is complete, the machine records the caller's message. When the caller hands up, the machine hangs up and shuts off. Place the following in the diagram : call detected, answer call, play announcement, record message, caller hangs up, announcement complete.*

Solution :

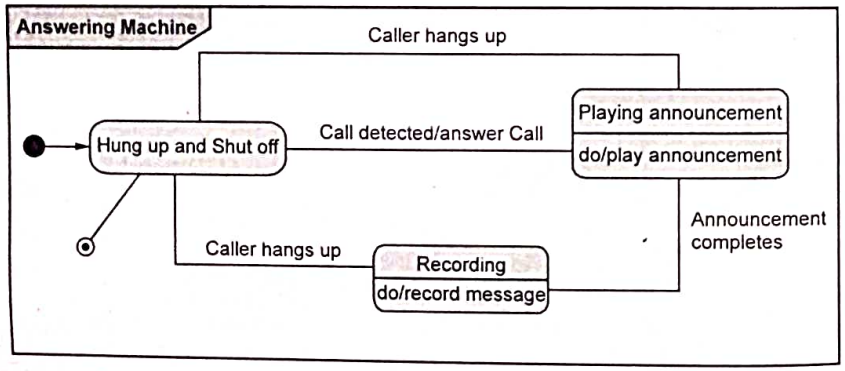


Fig. 2.15.5

Example 2.15.3 *Model a state transition diagram for the following scenario :*

Here is what happens in a microwave oven :

- *The oven is initially in an idle state with door open, where the light is turned on.*
- *When the door is closed it is now in idle but the light is turned off.*
- *If a button is pressed, then it moves to initial cooking stage, where the timer is set and lights are on and heating starts.*
- *At any moment the door may be opened, the cooking is interrupted, the timer is cleared and heating stops.*
- *Also while cooking, another button can be pushed and extended cooking state starts, where the timer gets more minutes. At any moment door can be opened here also.*
- *If the timer times out, then cooking is complete, heating stops, lights are off and it sounds a beep.*
- *When the door is open, again the oven is in idle state with the door open.*

AU : Dec.-17, Marks 13

Solution :

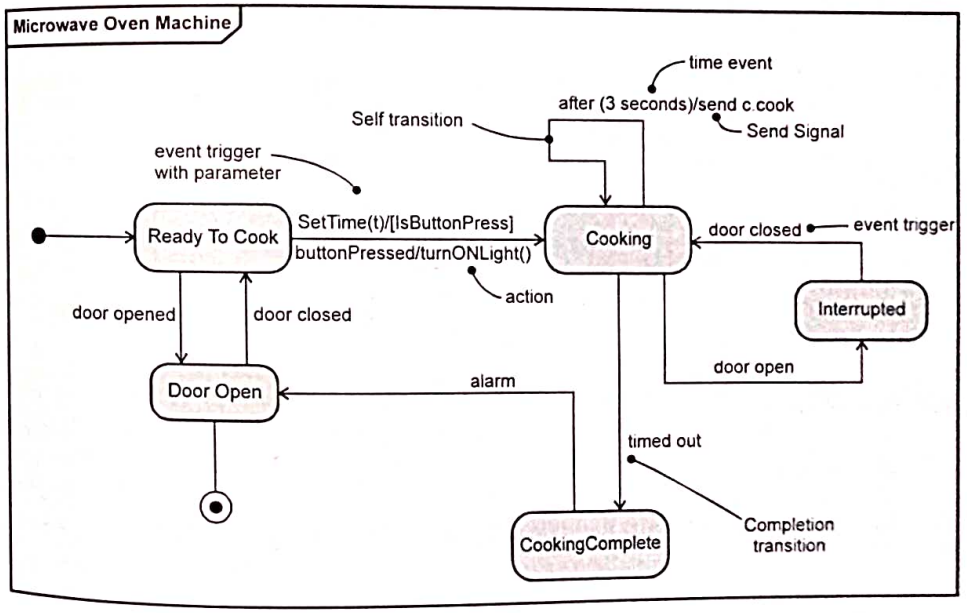


Fig. 2.15.6

Example 2.15.4 *Draw a state diagram for elevator.*

Solution :

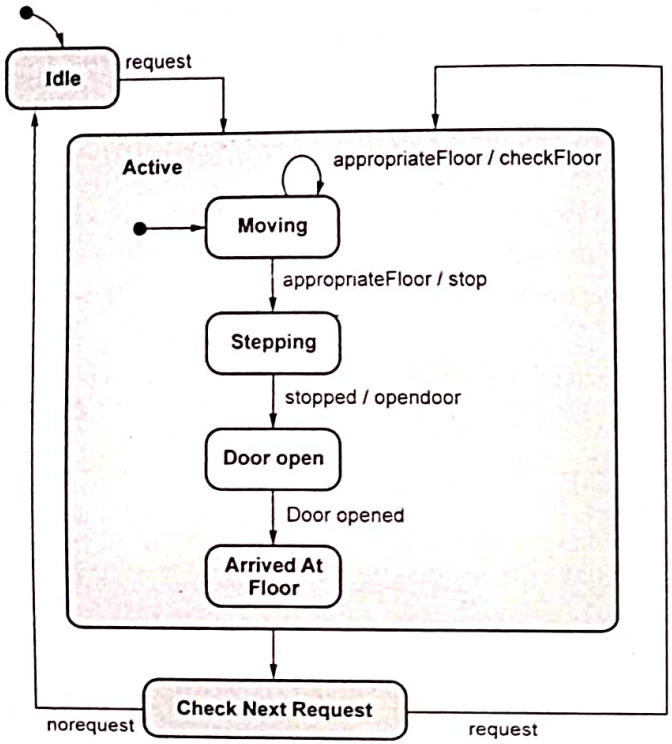


Fig. 2.15.7

Example 2.15.5 *Prepare an object oriented diagram for the controller in a home security system, which is responsible for monitoring various sensors around the perimeter of the house. Briefly explain.*

Solution : Fig. 2.15.8 represents the state diagram for home security system.

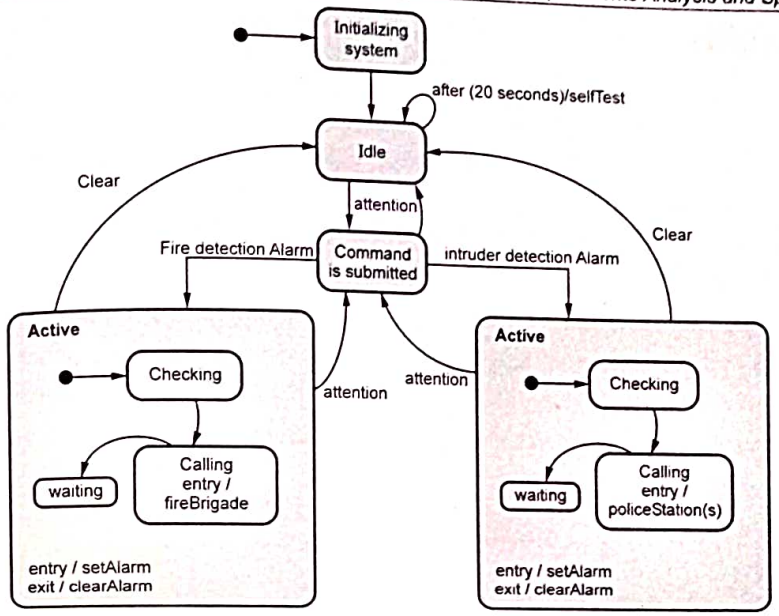


Fig. 2.15.8 Home security system

Example 2.15.6 *Draw a state diagram for a printer.*

Solution : The printer has two explicit states **on** and **off.** When it is **on** it undergoes through various substates such as **Idle, Working, Low tray, Empty tray** and **Full tray** states. When printer is in **off** state it is triggered to **on** state by pushing the button to on, similarly when printer is on then it can be brought to off state by pushing the button. (Refer Fig. 2.15.9)

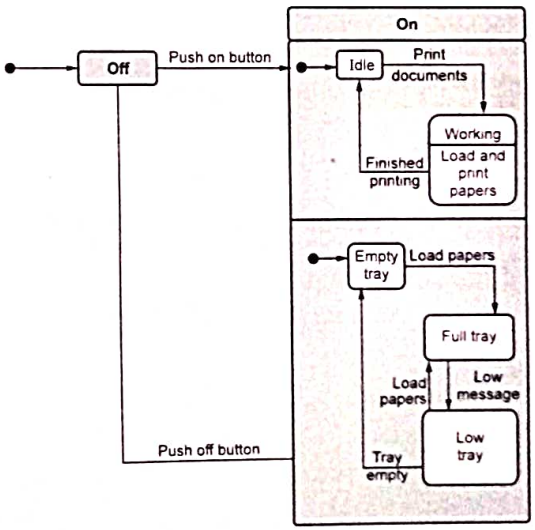


Fig. 2.15.9 State - machine for printer

Example 2.15.7 *A simple digital watch has a display and two buttons, button 'A' and 'B' to set it. The watch has two modes of display, 'display time' mode and 'set time' mode. In the display time mode, the watch displays the hours and minutes separated by a flashing colon. The set time mode has two sub modes, set hours and set minutes. The 'A' button selects modes. Each time it is pressed, the mode advances in a sequence : display time, set hours and set minutes. Within the sub modes, the B button advances the hours or minutes once each time it is pressed. Buttons must be released before they can generate another event. Prepare a state diagram of the watch.*

Solution :

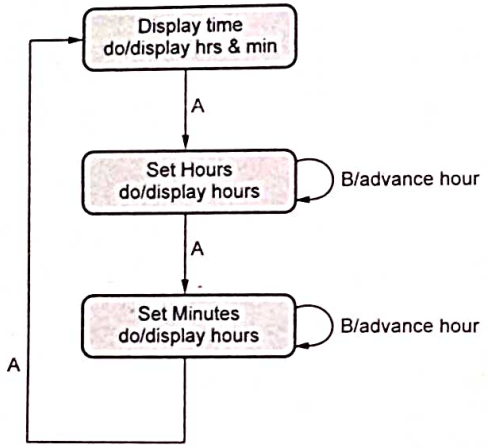


Fig. 2.15.10 Digital watch

Example 2.15.8 *Draw a state machine diagram for coffee vending machine.*

Solution :

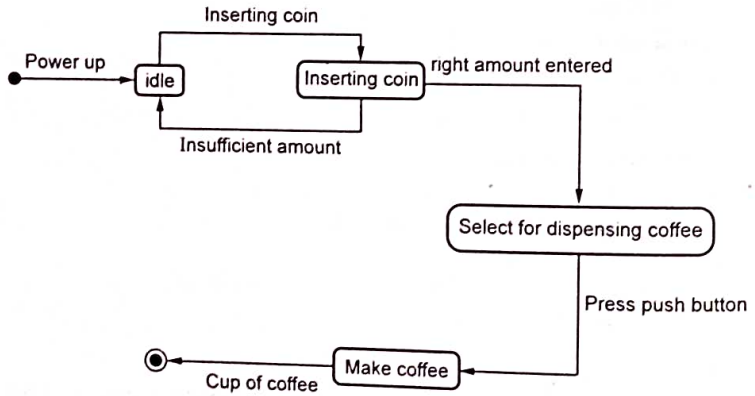


Fig. 2.15.11

2.15.4 When to use State Diagram ?

- State diagrams are used **to model the dynamic aspects** of a system. That means the event ordered behavior of any kind of object is represented using state diagram. The classes, interfaces, components and nodes can be represented in the state diagram.
- The state diagram can be used in context of the **system as a whole**. The state diagrams can also be attached to the use cases.
- The state diagrams can be used **to model reactive objects**. The reactive or event driven object is a kind of object whose behavior is characterized by the **responses to the events**. The reactive object is typically idle until it gets some event. When it receives some event that reactive object responds based on previous event.
- State diagram is used **to model various states of the objects**. These states can be the stable states or the transitive states transiting due to event triggering.
- The state diagrams are used **to model the actions** that occur on each state change.

Review Questions

1. *Write elaborately on UML state diagram and modeling.*
2. *Explain the state chart diagram with a suitable example. Also define its components and use.* **AU : Dec.-11, Marks 16**
3. *Explain UML state machine diagrams and modeling.* **AU : Dec.-13, May-14, Marks 16**
4. *What is the purpose of state chart diagram ? How to draw state chart diagram ? Explain.* **AU : Dec.-14, Marks 8**
5. *Write a short note on the state diagrams.* **AU : May-19, Marks 3**
6. *Give the meaning of event, state and transition.* **AU : May-11, Marks 2**
7. *State independent and state dependant objects.*
8. *How to model state dependant objects ?*

2.16 Functional Modelling

AU : May-16, Marks 8

- In function-oriented design, the design of software system is divided into various **interacting units**.
- Each unit then becomes a function. Thus system is designed from a **functional point of view**.
- This approach is more used in smaller systems but not preferred for larger systems.
- Various models that can be created in function-oriented design are -

  1. Data flow models 2. Data dictionaries 3. Structure charts

Difference between Function oriented design and Object oriented design

Sr. No.	Function oriented design	Object oriented design
1.	The problem is decomposed into functions and procedures.	The problem is decomposed into real-world entities as objects.
2.	Functions can be reused and combined to create more complex functionality.	Objects are self-contained entities that contain data and behavior.
3.	In function oriented design, state information is centrally located and several functions on this centralized data are defined.	In object oriented design, state information is shared among the objects.
4.	Functional-oriented design can be difficult to keep track of all the functions and how they interact with each other.	Object-oriented designs are easier to manage as the design is decomposed into objects and modules.
5.	Function-oriented design is associated with procedural programming like C.	Object-oriented design is associated with object oriented programming like C++, Java.
6.	The function-oriented design does not support inheritance and polymorphism.	The object-oriented design supports inheritance and polymorphism.
7.	The function oriented approach is simple to design.	Using this design approach bugs can be easily located and corrected without affecting the other module.
8.	For example - In railway reservation system the major functions will be get_personal_details(),reserve(),cancel(), compute_fair()	For example - To design a railway reservation system various classes could be-Class Passenger, the attributes are : name, age, sex, source, destination and the Operations are : get_personal_data(), reserve(),cancel(), Class Ticket, the attributes are : Ticket_ID, Source, destination, date, fair_amount and operation is : compute_fair()

Review Question

1. *Discuss the differences between object-oriented and function-oriented design.*

AU : May-16, Marks 8

2.17 Data Flow Diagram

AU : May-08,10,15,17,18,19, Dec.-13,15,16,17, Marks 16

The structured system analysis is a technique in which the system requirements are converted into specifications and then into computer programs, hardware configurations and related manual procedures.

- The structured analysis is mapping of problem domain to flows and transformations.
- The system can be modeled using :
  - Entity relationship diagram are used to represent the data model.
  - Data flow diagram which is used to represent the functional model.
- Data Flow Diagram (DFD) represents the flow of information for any process or system.

2.17.1 Rules for Designing DFD

1. No process can have only outputs or only inputs. The process must have both outputs and inputs.

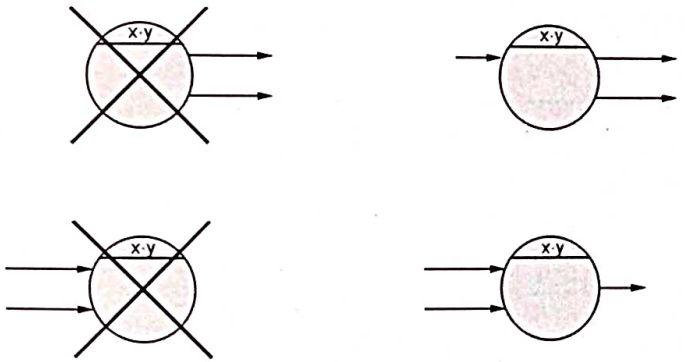


Fig. 2.17.1

2. The verb phrases in the problem description can be identified as processes in the system.
3. There should not be a direct flow between data stores and external entity. This flow should go through a process.

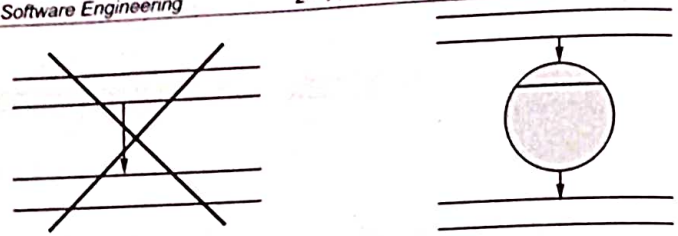


Fig. 2.17.2

Fig. 2.17.3

4. Data store labels should be noun phrases from problem description.
5. No data should move directly between external entities. The data flow should go through a process.

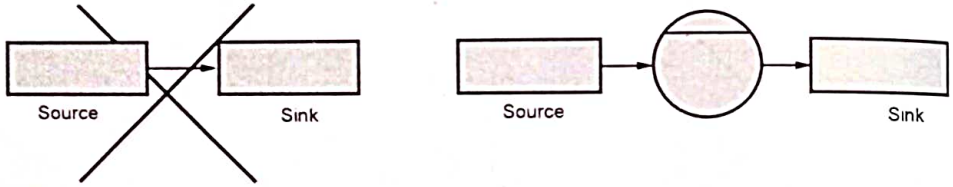


Fig. 2.17.4

6. Generally source and sink labels are noun phrases.
7. Interactions between external entities is outside scope of the system and therefore not represented in DFD.
8. Data flow from process to a data store is for updation/insertion/deletion.
9. Data flow from data store to process is for retrieving or using the information.
10. Data flow labels are noun phrases from problem description.

Example 2.17.1 *Design DFD for library management system for level 0 DFD and level 1 DFD.*

AU : Dec.-16, Marks 8

Solution : A **student** comes to a library for borrowing book. The student makes the book request by giving book title and author name. The student has to submit his library card to the library. Sometimes student may simply give topic and demand for a book (For example "just give me book on data structure"). The library information system maintains list of authors, list of titles, list of topics. This system also keeps record of topics on which books are available with the system. This system maintains information about shelf number on which books are located. Finally the list of demanded book should be displayed, on the console for ease of selection.

The first level of DFD can be

Solution :

In this DFD the whole system is represented with the help of input, processing and output. The input can be -

i) Student requests for a book hence **Book request.**

ii) To show identity of the student he/she has to submit his/her Library card, hence **Library card.** The processing unit can be globally given as

Library information system

The system will produce following outputs-

i) The demanded book will be given to student. Hence **Book** will be the output.

ii) The library information system should display **demanded book information** which can be used by customer while selecting the book.

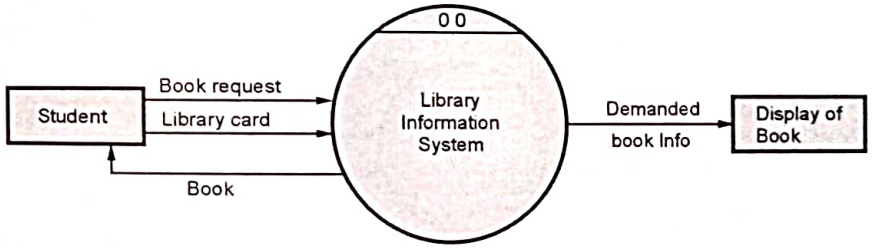


Fig. 2.17.5 Level 0 DFD (Context level DFD)

Level 1 DFD

In this level, the system is exposed with more processing details. The processes that need to carried out are -

i) Delivery of book ii) Search by topic

These processes require certain information such as List of Authors, List of Titles, List of Topics, the book selves from which books can be located. This kind of information is represented by **data store.**

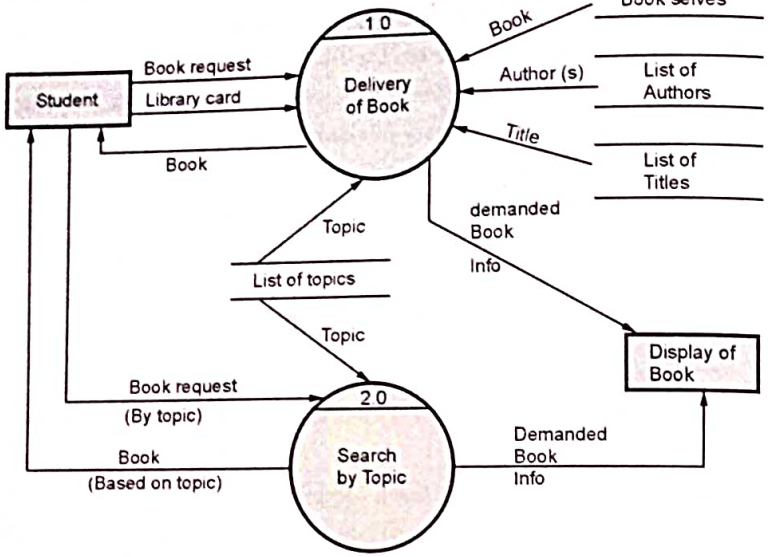


Fig. 2.17.6 Level 1 DFD

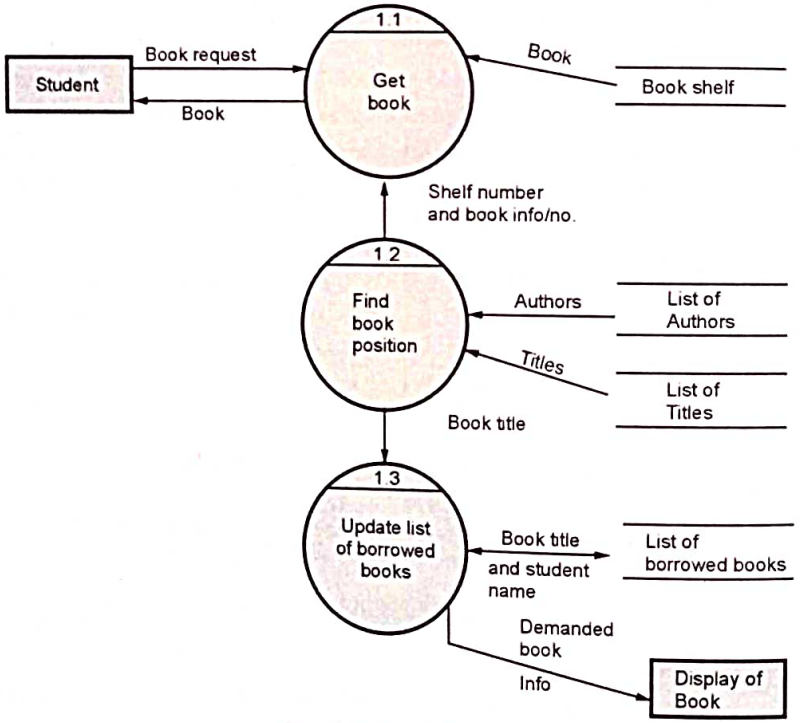


Fig. 2.17.7 Level 2 DFD

Example 2.17.2 *Draw use case and data flow diagrams for a restaurant system. The activities of the restaurant system are listed below. Receive the customer food orders, produce the customer ordered foods, serve the customer with their ordered foods, collect payment from customers, store customer payment details, order raw materials for food products, pay for raw materials and pay for labor.* **AU : Dec.-16, Marks 8, May-15, Marks 16**

Solution : A **customer** goes to a restaurant and orders for the food. The food order is noted down carefully and this order is sent to kitchen for preparing the required food.

This restaurant has to manage one housekeeping department which maintains **sold items** and **inventory data.** The daily information about sold items and inventory deplition amount is used to generate a **management report**. Finally this management report is given to **restaurant manager.**

In this level, the system is designed globally with input and output. The input to Food ordering system are -

1. As customer orders for the food. Hence food order is an input.

Level 0 DFD

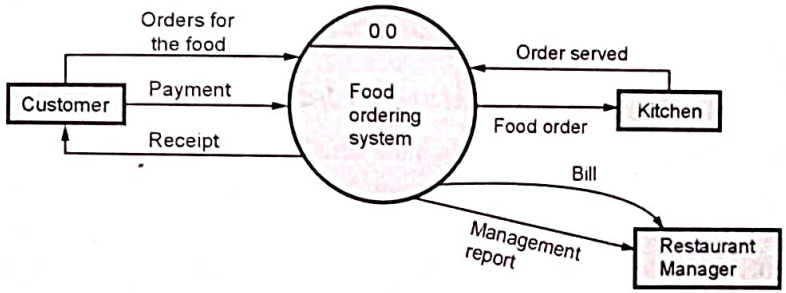


Fig. 2.17.8 Level 0 DFD (Context level DFD)

The output to food ordering system are -

1) Receipt.

2) The food order should be further given to kitchen for processing the order.

3) Bill and management report is given to restaurant manager.

Level 1 DFD

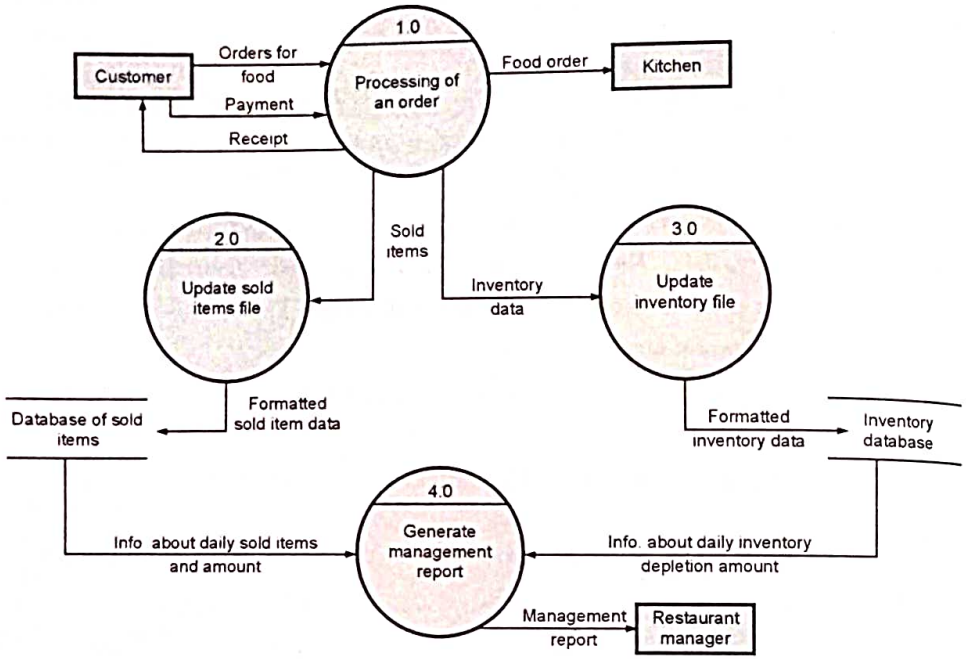


Fig. 2.17.9 Level 1 DFD

In this level, the bubble 0.0 is shown in more detail by various processes. The process 1.0 is for processing an order. And processes 2.0, 3.0 and 4.0 are for housekeeping activities involved in food ordering system. To create a management report there should be some information of daily sold items. At the same time inventory data has to be maintained for keep track of 'instock' items. Hence we have used two **data stores** in this DFD -

1. Database of sold items
2. Inventory database

Finally management report can be prepared using daily sold details and daily inventory deplition amount. This management report is given to restaurant manager.

In this DFD we will elabortate "processing of order".

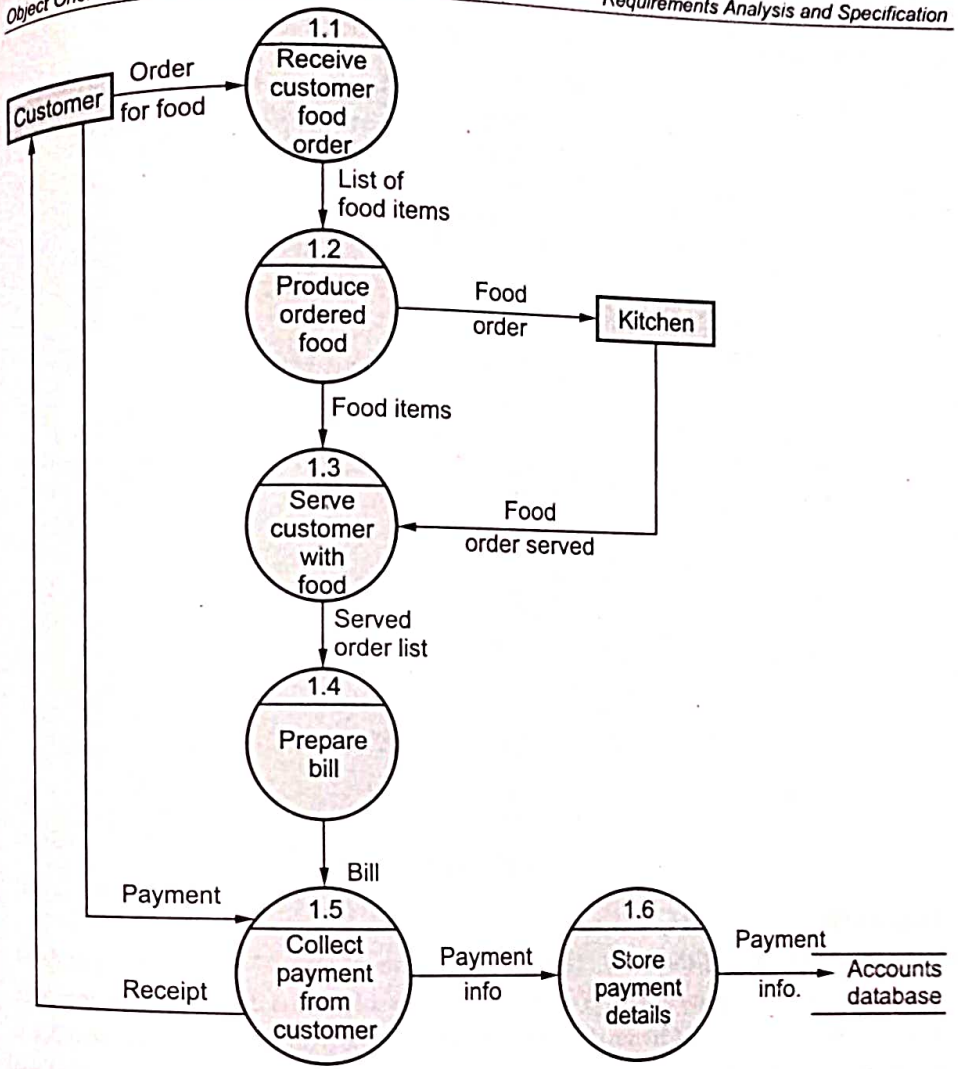


Fig. 2.17.10

The DFA for "update for inventory file " is shown in detail as follows -

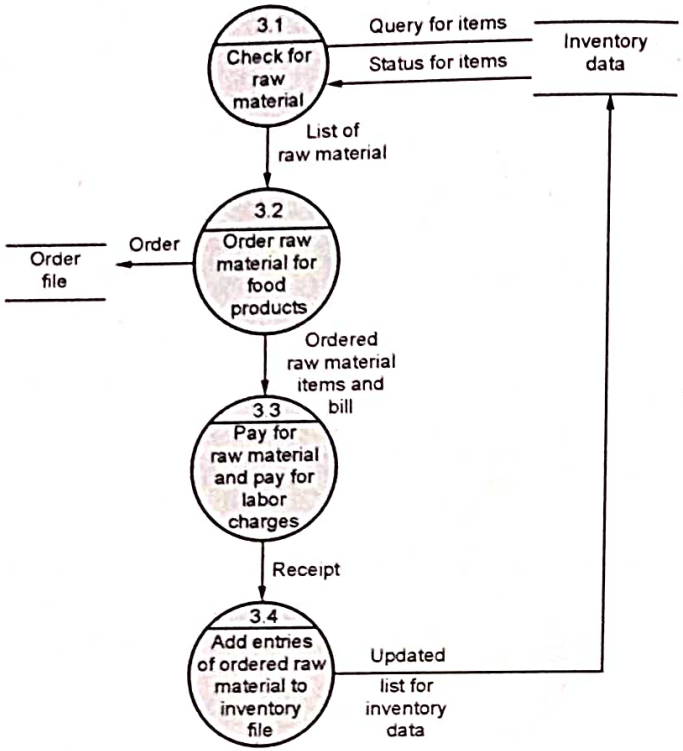


Fig. 2.17.11

Level 2 DFD

In this DFD we will elaborate "Generate management report" activity in more detail. For generating management report we have to access sold items data and inventory data. Then aggregate both solid items data and inventory data. Total price of each item has to be computed. Then from these calculation a management report has to be prepared and given to the restaurant manager. These details can be shown in this DFD.

(See Fig. 2.17.12 on next page.)

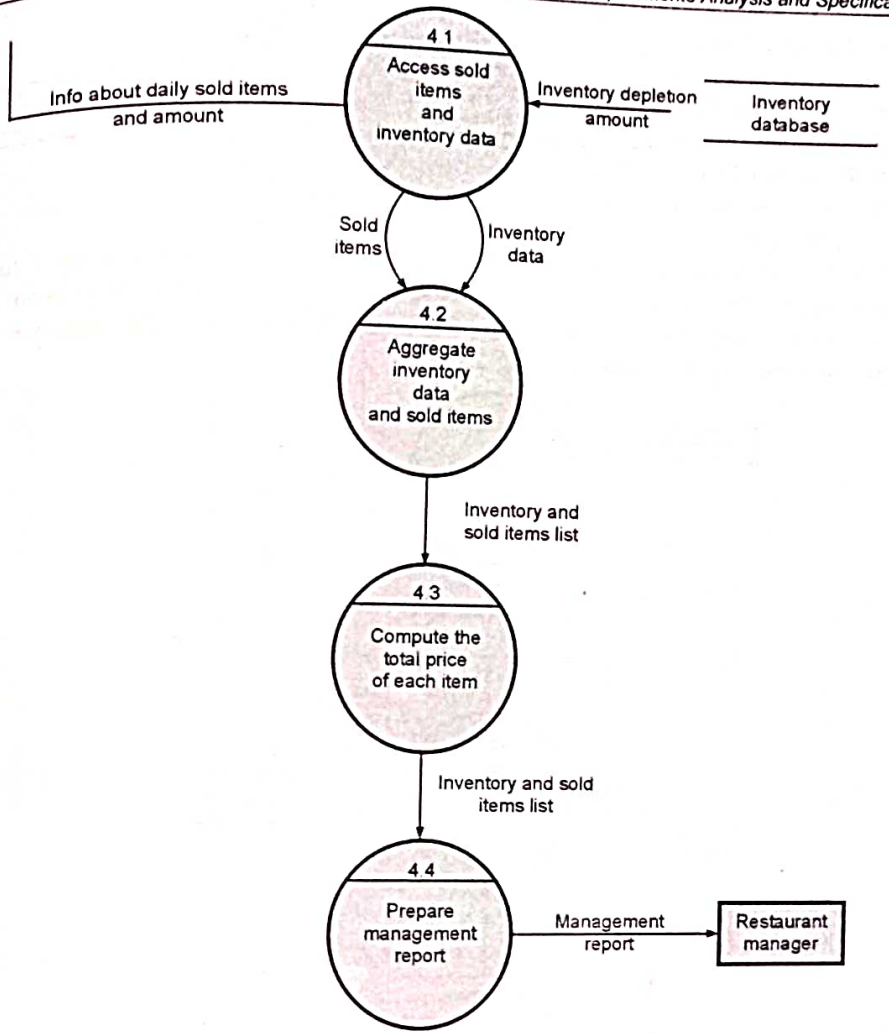


Fig. 2.17.12 Level 3 DFD

Example 2.17.3 *Prepare a DFD for a Books order processing system.*

Solution : The **customers** in this system are book sellers who do not make a stock of books. As the **orders** come the books are demanded from **publishers** directly.

As per the problem description it is clear that the input to this system is 'customer' who places an order and output will be purchase order given to publisher.

Now, we will do more detailing for order processing when an order is received by the system, it will be first verified using the books database. Credit rating will be decided by checking customer details (i.e. whether the customer is regular customer or whether he is

new). For submitting a batch of orders we have to maintain pending orders list as well. There may be the order for books which are published by different publishers, in such a case database for different publishers need to be maintained by the system. This list of purchase orders is maintained by the system and then after verifying the shipment a shipping note will be given to the customer. The level 1 decomposition is as shown in Fig. 2.17.14.

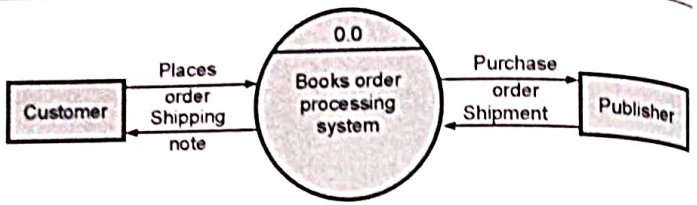


Fig. 2.17.13 Level 0

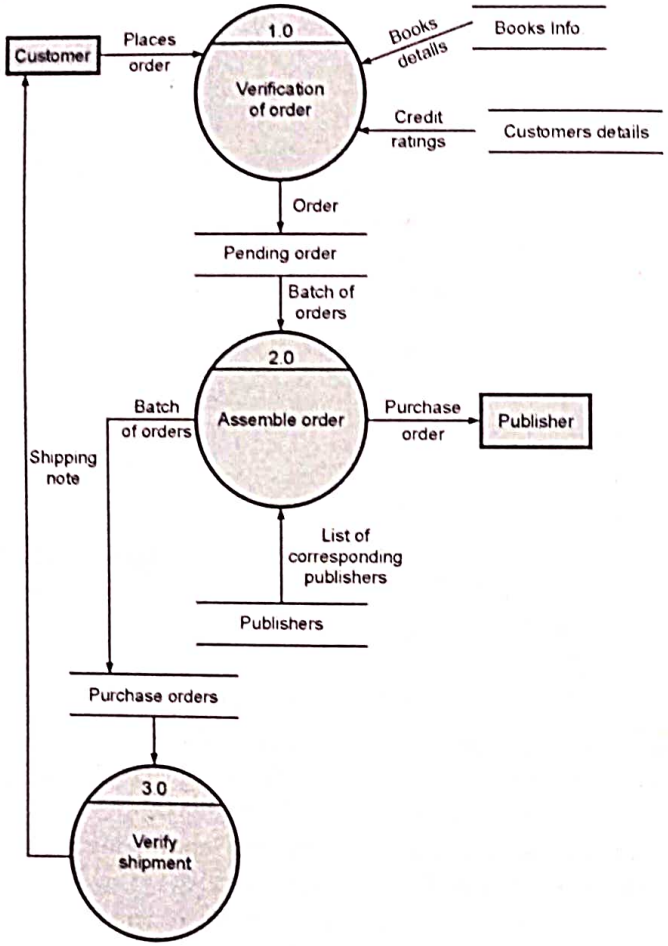


Fig. 2.17.14 Level 1

We can further decompose the system by elaborating the process **Assemble order.** In assemble order we

- First get details of total copies per title.
- Then prepare purchase order accordingly for corresponding publishers.
- Send these purchase orders to publishers.
- These purchase order details should be stored in the system. Let us draw level 2 DFD for these details.

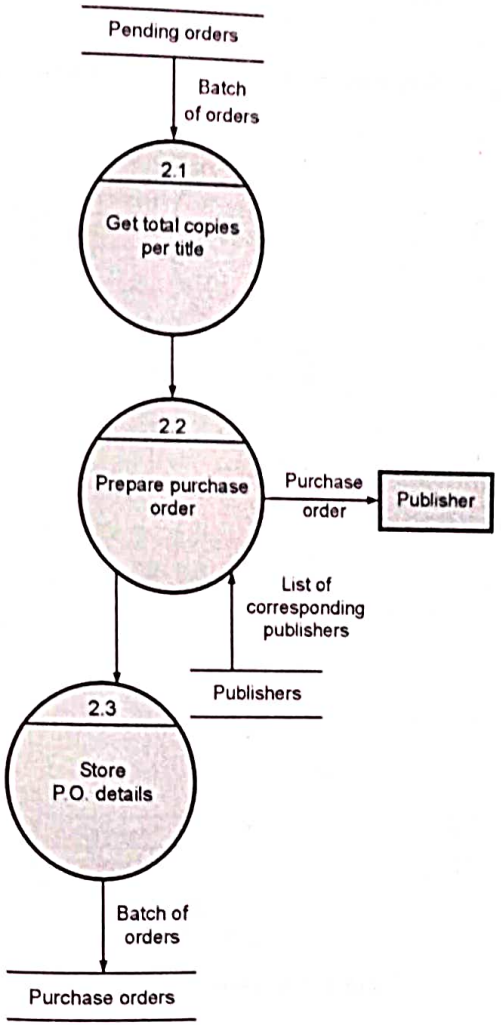


Fig. 2.17.15 Level 2

Example 2.17.4 *Design DFD for Hotel reservation system.*

Solution : In a hotel reservation system, a **customer** can make online booking for a hotel, by specifying the accomodation requirements such as type of room (AC/Non AC/One bed/Two bed), total number of rooms, duration of stay. The system then **selects** a suitable hotel as per customer's requirements. If such a hotel is found then the **availability** of rooms in that hotel is **checked.** The **charges are calculated** for the selected requirement and these are acknowledged to the customer. If the customer is satisfactory about the selection made by the system then he **confirms** the reservation.

The system then gives these booking details to the corresponding **hotel.** Design DFD for this system.

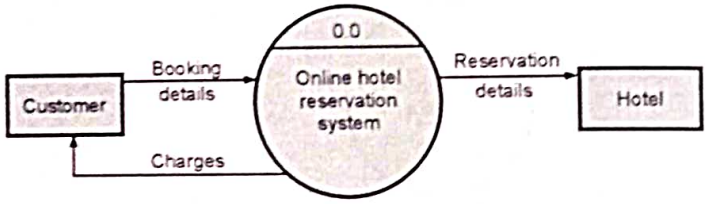


Fig. 2.17.16 Level 0 DFD

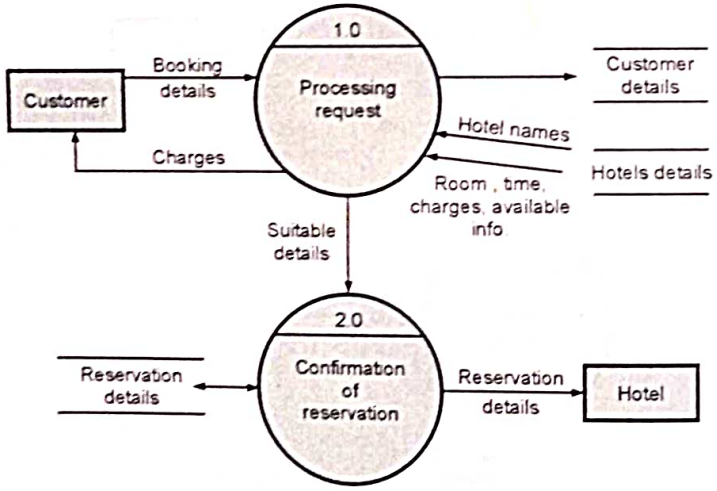


Fig. 2.17.17 Level 1 DFD

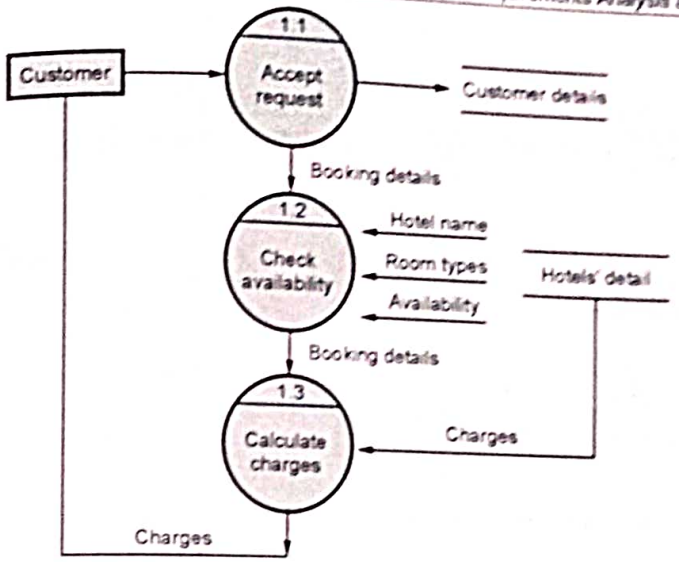


Fig. 2.17.18 Level 2 DFD

Example 2.17.5 *Tamil Nadu Electricity Board (TNEB) would like to automate its billing process. Customers apply for a connection-(domestic/commercial). EB staff take readings and update the system each customer is required to pay charges bi-monthly according to the rates set for the type of connection. Customers can choose to pay either by cash/card. A bill is generated on payment. Monthly reports are provided to the EB Manager.*

i) Give a name for the system. (Mark 1)
ii) Draw the Level - 0 DFD (Context Flow Diagram) (Marks 5)
iii) Draw the Level - 1 DFD. (Marks 10) **AU : Dec.-13. Marks 16**

Solution : i) Electricity Bill Management System

ii)

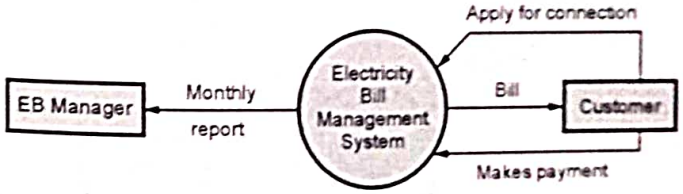


Fig. 2.17.19 Level 0 DFD

iii)

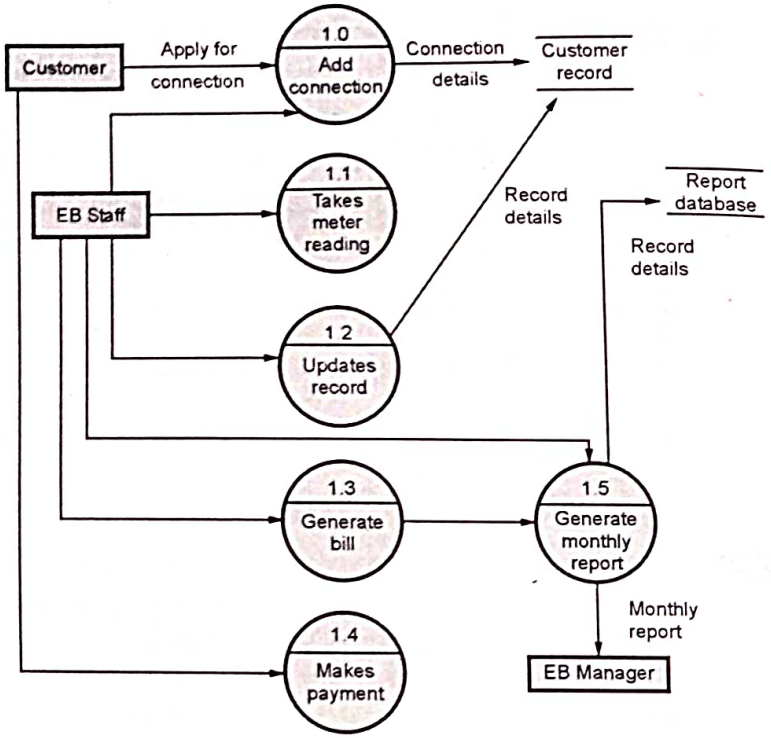


Fig. 2.17.20 Level 1 DFD

Example 2.17.6 *Develop a dataflow diagram for burglar alarm system along with entity relationship diagram.* **AU : May-10, Marks 8**

Solution : 1) DFD :

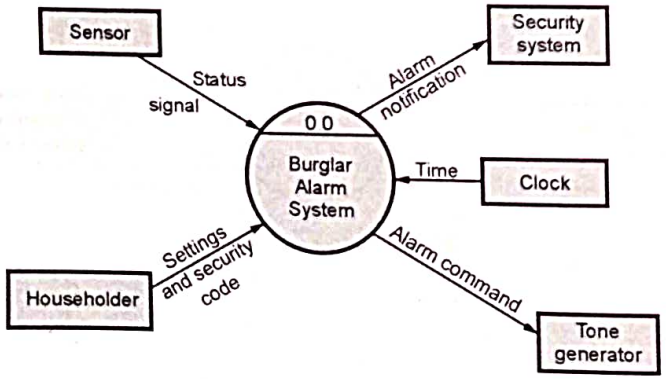


Fig. 2.17.21 Context diagram

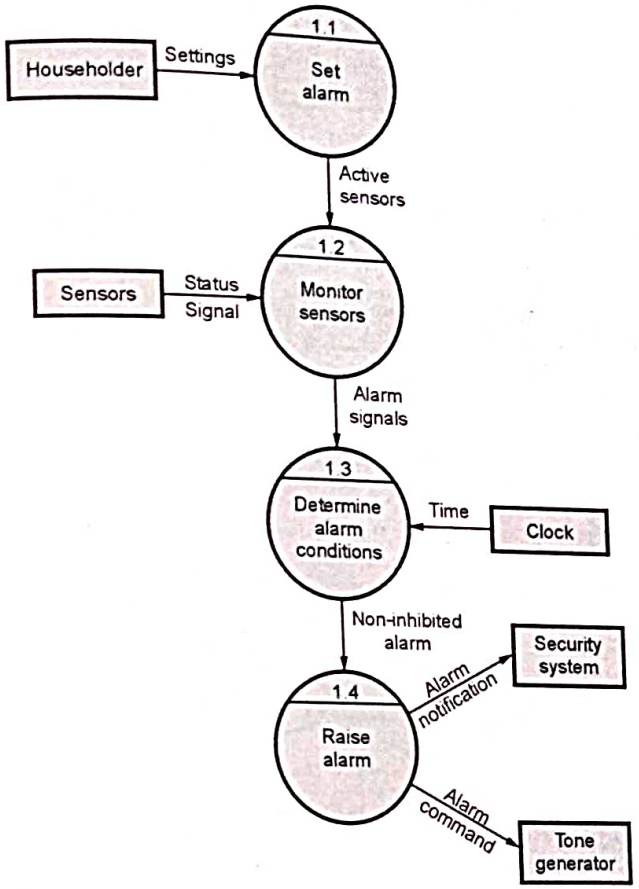


Fig. 2.17.22 Level 1 DFD

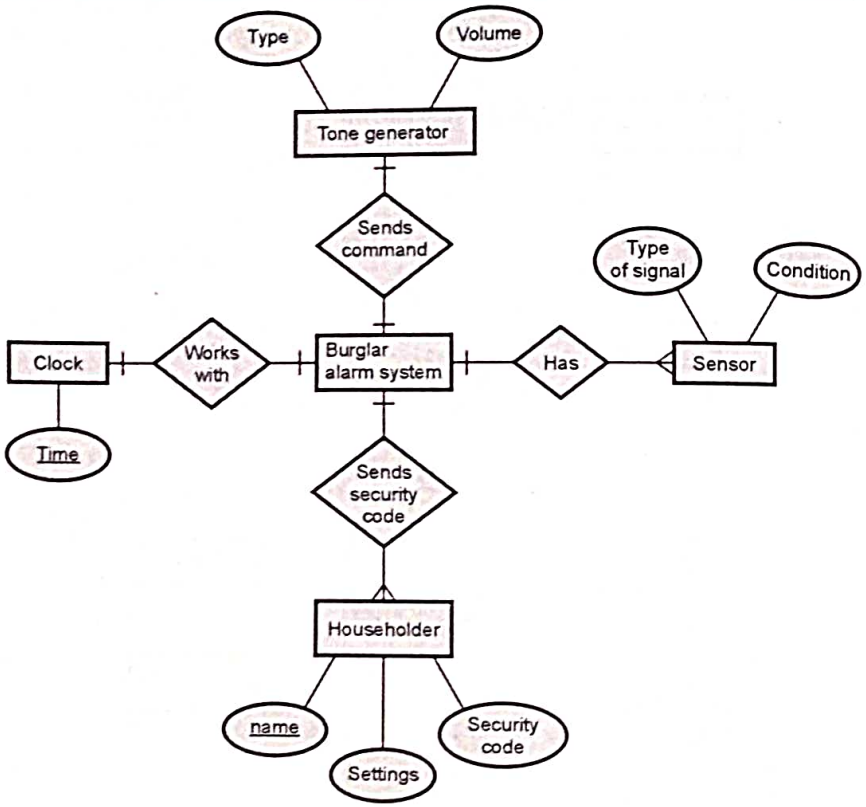


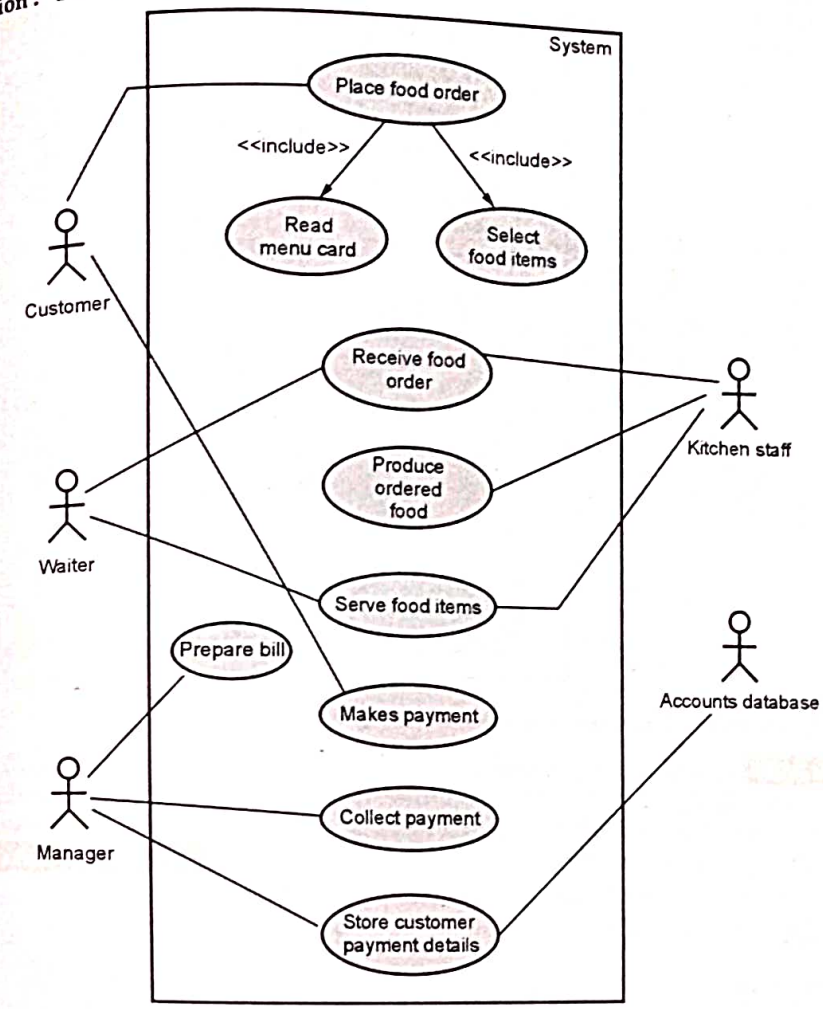
Fig. 2.17.23 ER diagram

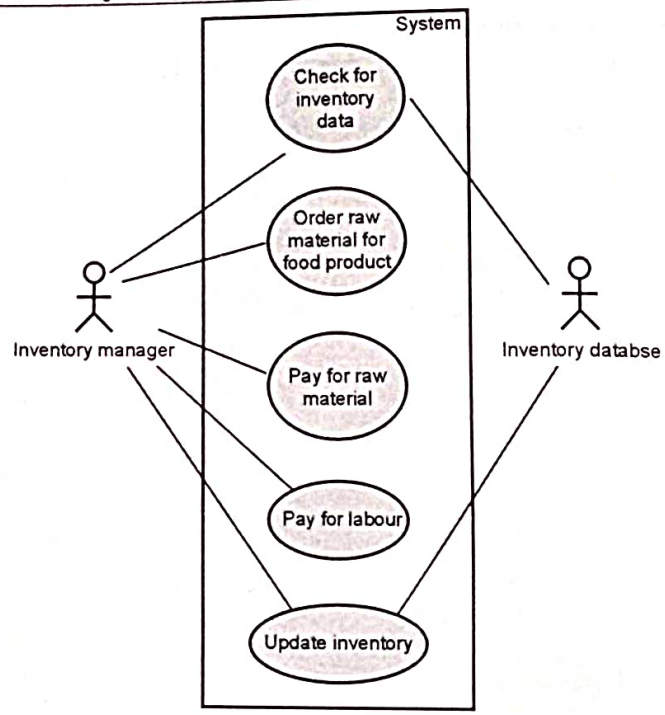
Example 2.17.7 *Draw the use case and data flow diagrams for a restaurant system. The activities of restaurant system are listed below -*

Receive the customer food orders, produce the customer ordered foods, serve the customer with their ordered foods, collect payment from customers, store customer payment details, order raw materials for food products, pay for raw materials and pay for labour.

AU : May-15, Marks 16

Solution : Part I : Use case diagram :





Part II : Data flow diagram : Refer example 2.17.2.

Example 2.17.8 *Consider an online railway reservation system, which allows the user to select route, book/cancel tickets using net banking/credit/Debit cards. The site also maintains the history of the passengers. For the above system, list and draw the use case scenario and model the above specification using data flow diagram.* **AU : Dec.-15, Marks 16**

Solution : The use cases for the given railway reservation system are -

1. Logs in
2. Select route
3. Book or cancel tickets
4. View status
5. Logs out

The use case diagram is as follows -

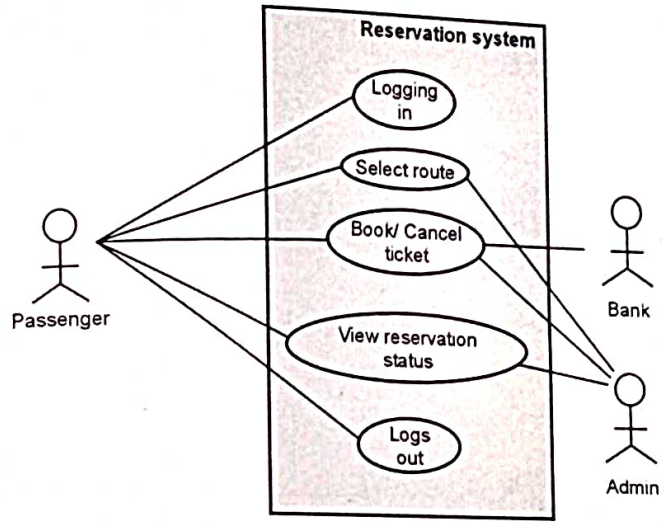


Fig. 2.17.24 Use case diagram

The level 0 and level 1 DFD can be as shown below –

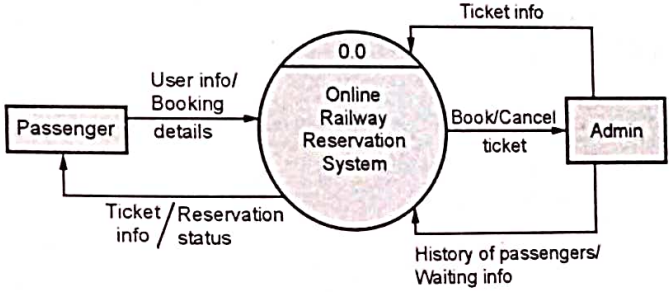


Fig. 2.17.25 Level 0 DFD

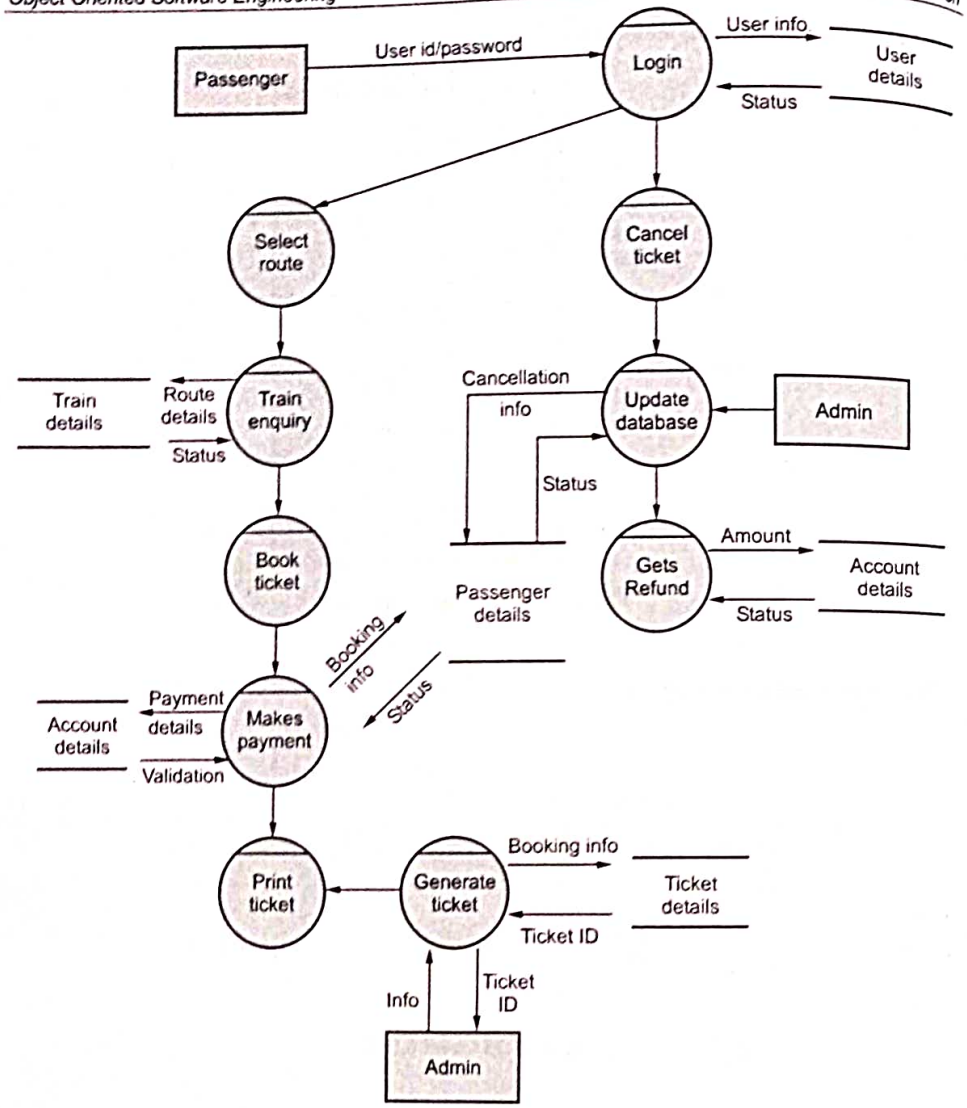


Fig. 2.17.26 Level 1 DFD

Example 2.17.9 *Consider an online book stores. It accepts individual/bulk orders, process payments, triggers delivery of the books. Some of the major features of the system include :*

- *Order books*
- *User friendly online shopping cart function*
- *Create, view, modify and delete books to be sold*
- *To store inventory and sales information in database*
- *To provide an efficient inventory system*
- *Register for book payment options*
- *Request book delivery*
- *Add a wish list*
- *Place request for books not available*
- *To be able to print invoices to members and print a set of summary reports*
- *Internet access.*

Analyse the system using the context diagram and level 1 DFD for the system. Explain the components of DFD. **AU : May-17, Marks 15**

Solution :

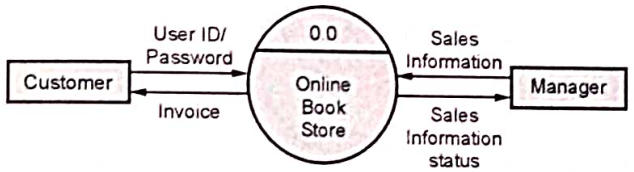


Fig. 2.17.27 Context level DFD

Refer Fig. 2.17.28 on next page.

Example 2.17.10 *Consider the process of ordering a pizza over the phone. Draw the use case diagram and also sketch the activity diagram representing each step of the process, from the moment you pick up the phone to the point where you start eating pizza. Include activities that others need to perform. Add exception handling to the activity diagram you developed. Consider at least two exceptions (e.g. delivery person wrote down wrong address, deliver person brings wrong pizza).* **AU : Dec.-17, Marks 13**

Solution : Use case diagram

Refer Fig. 2.17.29 on page 2 - 165.

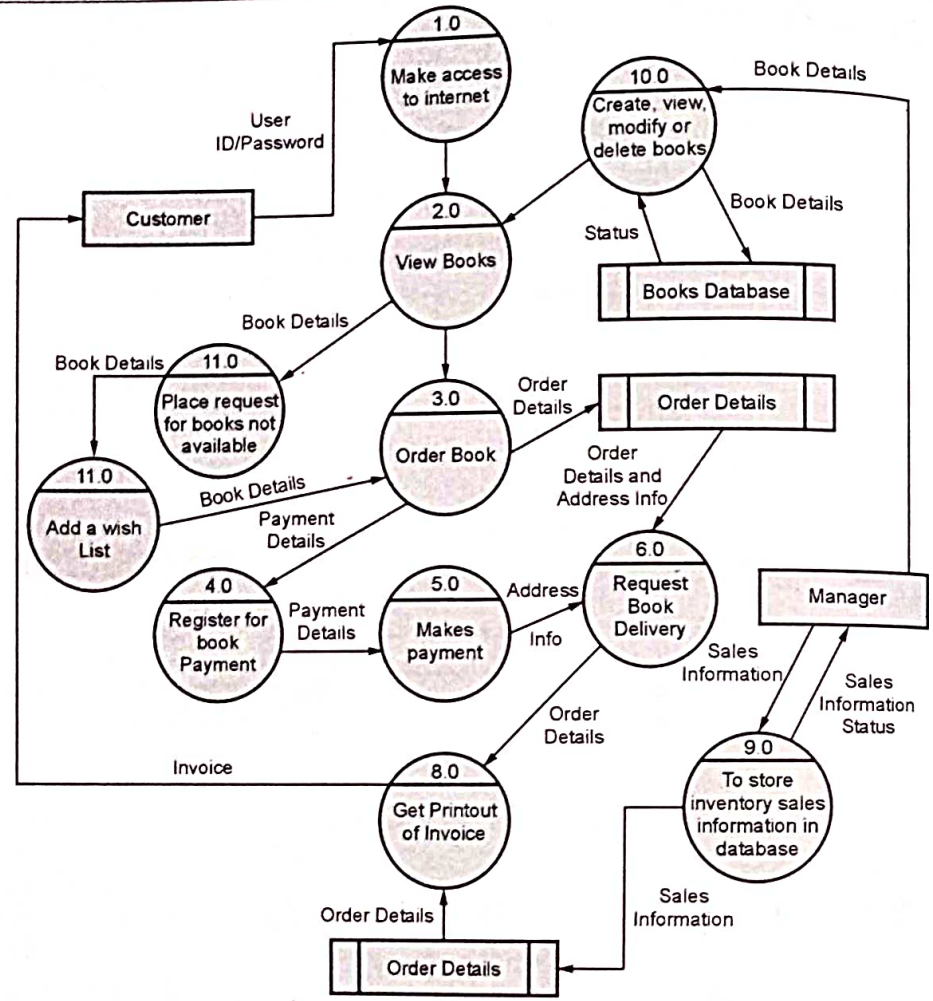


Fig. 2.17.28 Level 1 DFD

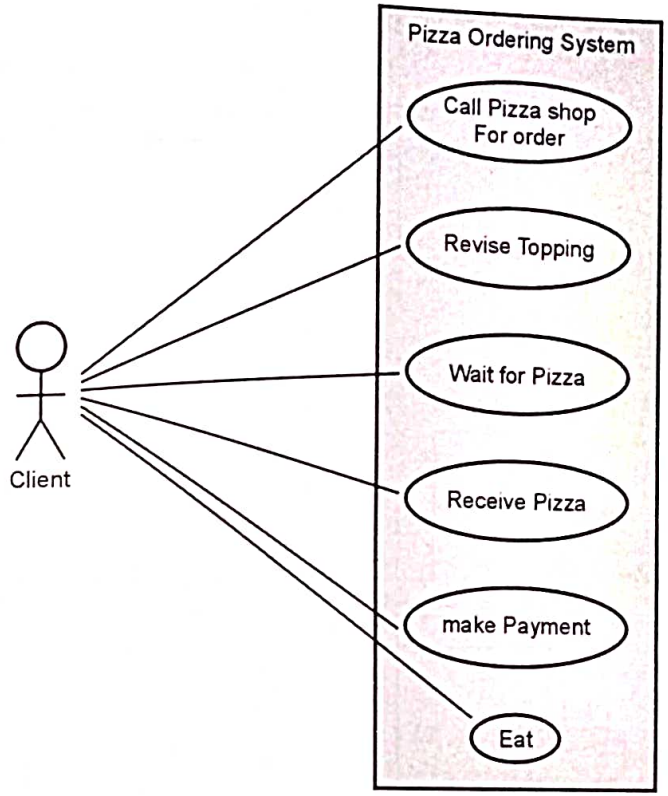


Fig. 2.17.29 Use case diagram for pizza ordering system

Activity diagram

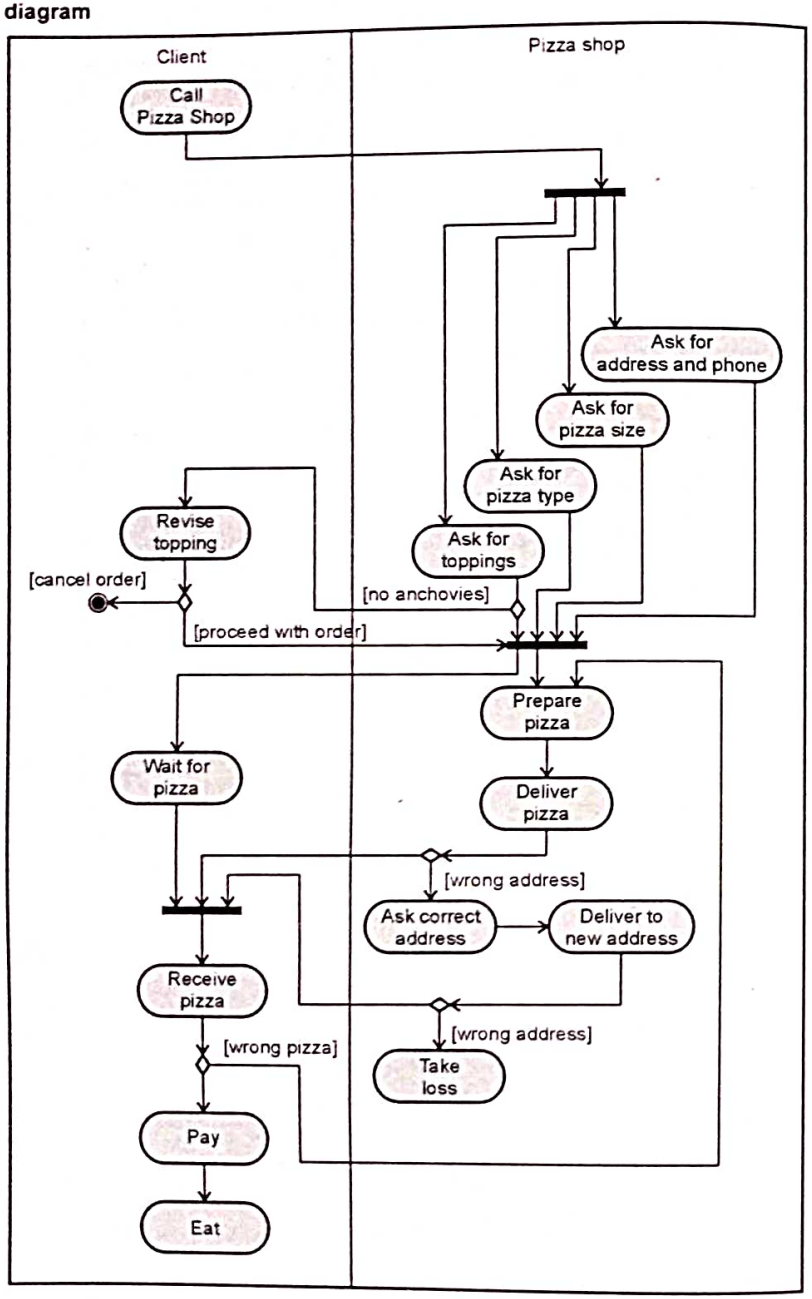


Fig. 2.17.30

Example 2.17.11 *What is the purpose of DFD ? What are the components of DFD ? Construct DFD for the following system : An on-line shopping system for XYZ provides many services and benefits to its members and staffs. Currently, XYZ staffs manually handle the purchasing information with the use of basic office software, such as Microsoft Offices Word and Excel. It may results in having mistakes easily and the process is very inconvenient. XYZ needs an online shopping system has five key features :*

i) to provide the user friendly online shopping cart function to members to replace hardcopy ordering form ;

ii) to store inventory and sales information in database to reduce the human mistakes, increase accuracy and enhance the flexibility of information processing ;

iii) to provide an efficient inventory system which can help the XYZ staffs to gain enough information to update the inventory ;

iv) to be able to print invoices to members and print a set of summary reports for XYZ's internal usage ;

v) to design the system that is easy to maintain and upgrade. **AU : May-18, Marks 15**

Solution : Level 0 DFD

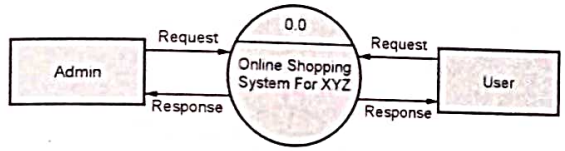


Fig. 2.17.31

Level 1 DFD

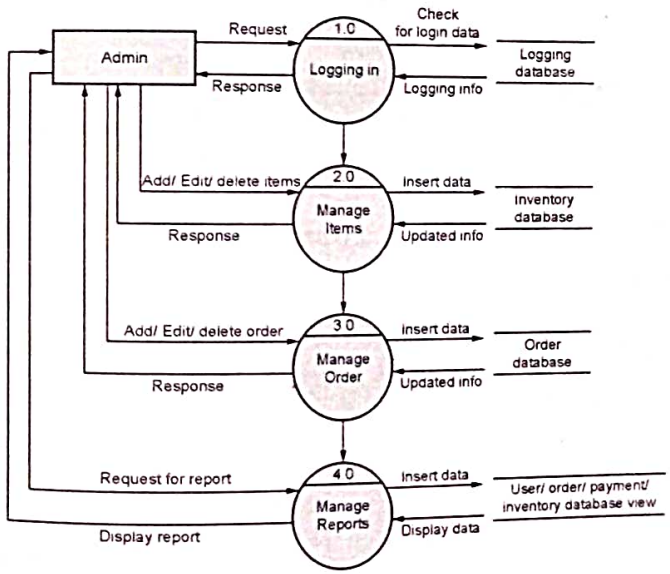


Fig. 2.17.32

Level 2 DFD : For manage order

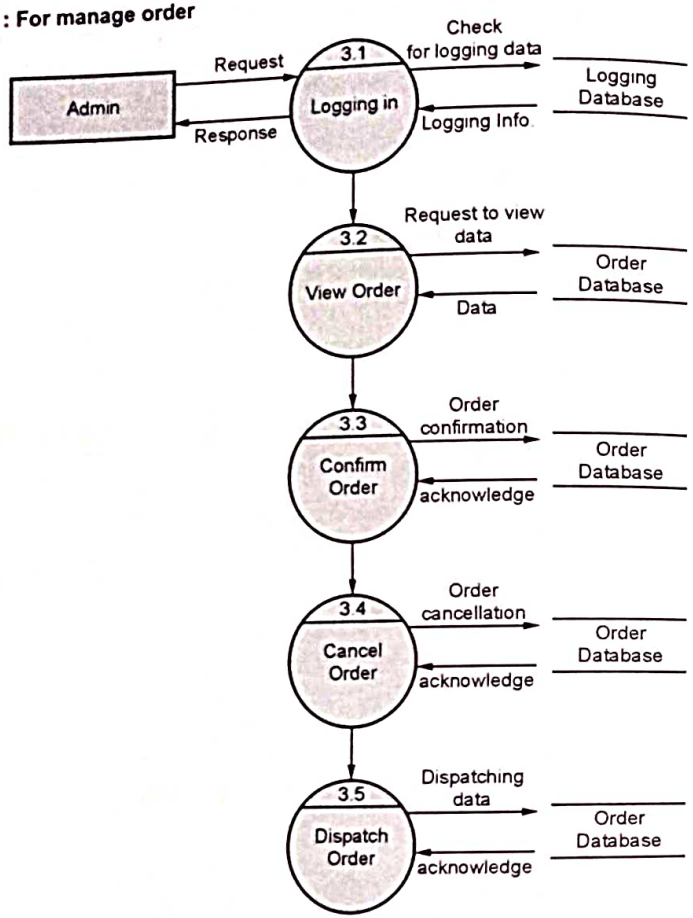


Fig. 2.17.33

Level 2 DFD : For manage items

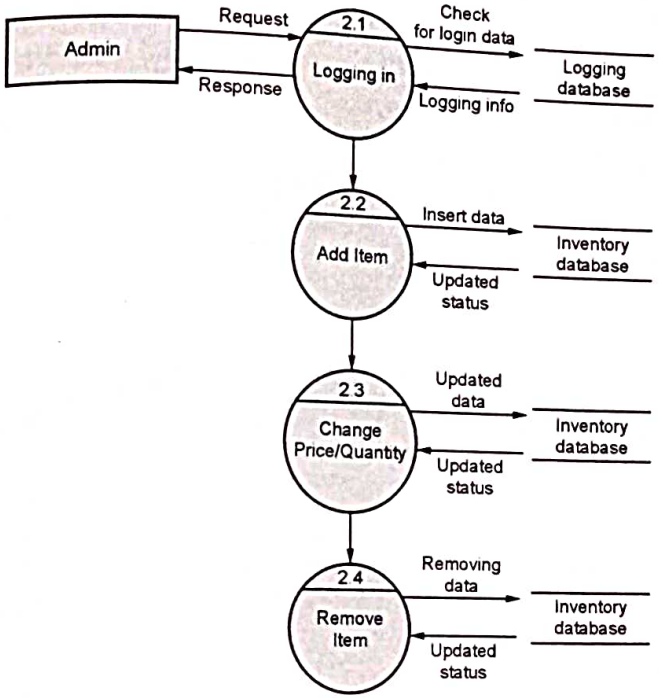


Fig. 2.17.34

Level 1 DFD : From user's perspective

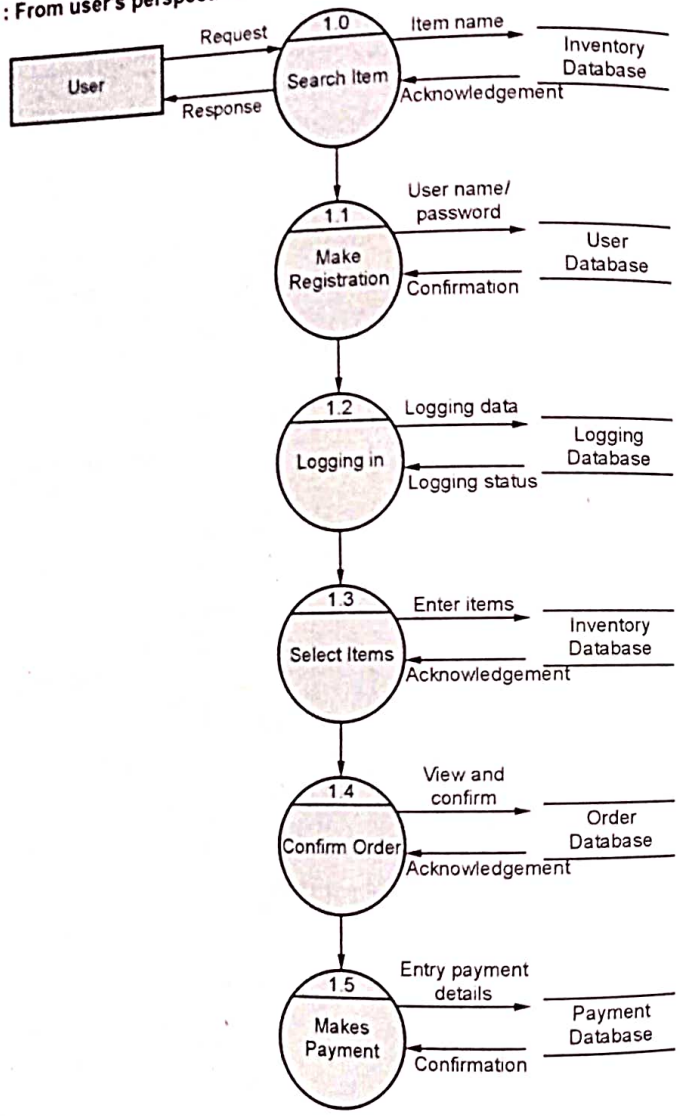


Fig. 2.17.35

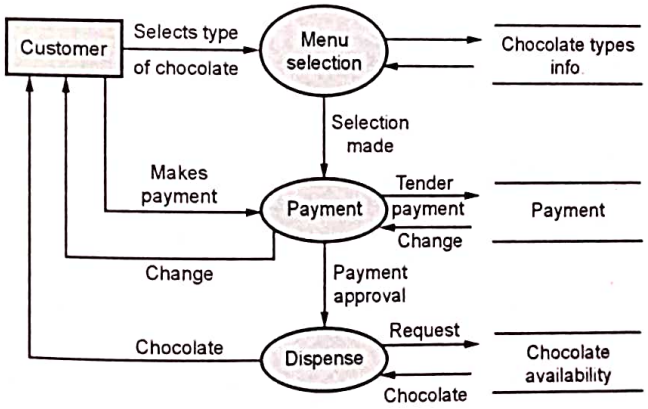
Example 2.17.12 *i) Draw the Level 0 and Level 1 data flow diagram for the following system.*

ii) Identify entities in the system and draw a diagram showing the relationship between entities.

The Chocolate Vending Machine (CVM) system requirements are as follows : The CVM dipenses chocolates : 1) Very large Chocolates (VC) at ₹15, 2) Large Chocolates (LC) at ₹10 and 3) a Small Chocolates (SC) at ₹5. The vending machine only. deals in coins. The CVM gives the proper change after the product selection is made. The CVM must check the amount deposited by the customer. The vending machine operates in the following Way. (A) The CVM remains idle until a customer or owner begins to interact with the machine. When a selection button is pressed the VCM indicates the required amount (₹15/-, ₹10/-, ₹5). (B) If the full amount needed has been deposited then dispense the proper chocolate and display : Thank You! (C) If an insufficient amount (possibly zero) has been deposited then display : remaining amount needed. (D) If an over amount has been deposited them dispense the proper candy and change arid display : Thank You! **AU : May-19, Marks 8 + 7**

Solution :

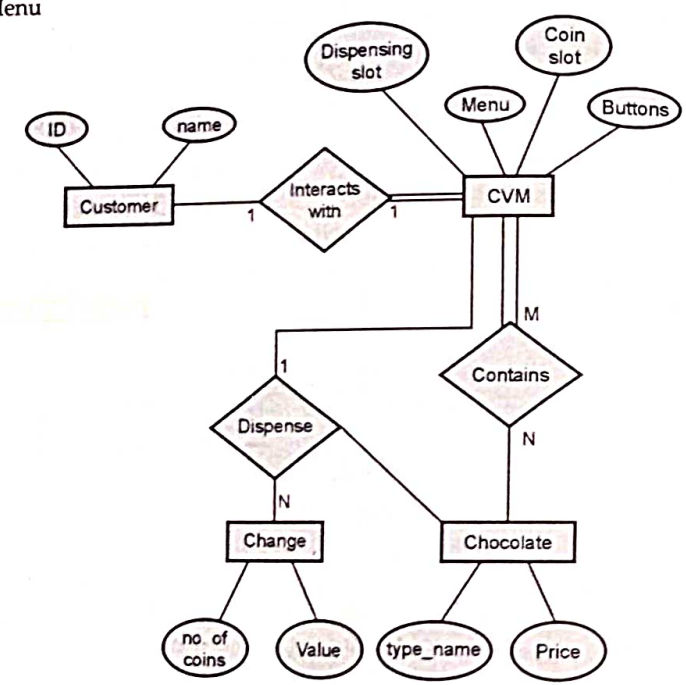
Level 0 DFD



Level 1 DFD

Entities in the system are :

1) Customer
2) CVM
3) Chocolate
4) Menu



2.17.3 Difference between Structured Analysis and Object Oriented Analysis

Structured analysis	Object oriented analysis
In structured analysis the process and data are separately treated.	In object oriented analysis the process(methods) and data are encapsulated in the form of object.
Various models that are getting developed in structured analysis are DFD, ER diagram and Data dictionary.	Various models that are getting developed in object oriented analysis are class diagram, use case diagram, activity diagram and deployment diagram.
Simpler to design.	Modular systems can be developed using this kind of analysis.
It is not suitable for large and complex projects.	For large and complex system object oriented approach is more suitable.

Review Question

1. *Describe primary differences between structured analysis and object oriented analysis.* **AU : May 08, Marks 6**

2.18 Two Marks Questions with Answers

Q.1 Write distinct steps in requirement engineering process. **AU : IT, May-06**

Ans. : The distinct steps in requirement engineering process are

1. Requirements elicitation.
2. Requirements analysis.
3. Requirements validation.
4. Requirements management.

Q.2 Why SRS must be traceable? What is traceability requirement ? **AU : IT, Dec.-05**

Ans. : There may be chances of having new requirements during development of the software. Traceability is concerned with relationship between requirements their sources and the system design. Thus change in requirement is manageable if SRS is traceable.

The traceability requirement is the relationship between dependent requirements.

Q.3 What are non-functional requirements for a software ? **AU : IT, May-03, May-05, Dec.-19**

Ans. : The non-functional requirements define system properties and constraints. Various properties of a system can be reliability, response time, storage requirements, and constraints of the system can be input and output device capability, system representations.

Q.4 What is the outcome of feasibility study ? **AU : IT, Dec.-03**

Ans. : The feasibility study decides whether the proposed system is worthwhile or not. The outcome of feasibility study is the results obtained from following questions :

1. Whether system contributes to organizational objectives ?
2. Whether the system can be engineered? Is it within the budget ?
3. Whether the system can be integrated with other existing system ?

After performing the feasibility study a feasibility study report is prepared.

Q.5 Differentiate data flow diagram and state transition diagram. **AU : IT, Dec.-06**

Ans. :

Data flow diagram	State transition diagram
Data flow diagram is a graphical representation for representing the information flow and the transforms that are applied as data move from input to output.	State transition diagram is a graphical representation for representing the behaviour of a system by depicting its states and the events that cause the system to change state.
The data flow diagram is a collection of process, data store, flow of data (transitions) and external entity.	The state transition diagram is a collection of states and events.
The data flow diagrams are used to represent the system at any level of abstraction, and the increasing levels are used to expose more and more functionalities in the system.	The state transition diagrams are typically drawn at single level. They are intended to expose the overall behaviour of the system.

Q.6 Why it is so difficult to gain a clear understanding of what customer wants ?

AU : CSE, Dec.-07

OR Write a note on what are the difficulties in elicitation, requirement elicitation.

AU : May-17, Marks 5

OR Requirements analysis is unquestionably the most communication intensive step in the software engineering process. Why does the communication path frequently breaks down ? **AU : May-16, Marks 8**

Ans. : Following are the reasons for : why it is difficult to understand customer wants -

1. Customer sometimes is unable to specify the **scope of the project**. Sometimes customers specify too many technical details and this may increase the confusion.
2. There is difficulty in **understanding the problem**. Sometimes customer could not decide what are their needs and wants. Sometimes they have got poor understanding of capabilities and limitations the existing computing environment.

   Sometimes customers find it difficult to communicate with the system engineer about their needs. Sometimes customers may have got some conflicting

requirements. This ultimately results in specifying ambiguous requirements.

3. As project progresses the needs or requirements of the customers changes. This creates a problem of volatility.

Q.7 Identify ambiguities and omissions in the functional requirements. What questions would you ask to clarify these functional requirements ? **AU : IT, May-08**

Ans. : Following are the questions that must be asked to clarify the functional requirements -

1. Is the requirement specification clear ? Can it be misinterpreted ?
2. What are the sources of requirements ? Is the final requirement specification as per the original source ?
3. Is there any requirement violation for domain constructs ?
4. What are the other related requirements of the current requirement ? And are they cross verified ?
5. For tracing the requirement is there any system model ?
6. Is the requirement testable ?
7. Has an index specification created for the requirements ?
8. Is the requirement satisfying the system objective.

Q.8 List out requirements engineering. **AU : CSE, May-09**

Ans. : Various activities in requirement engineering are -

1. Requirements elicitation 2. Requirements analysis
3. Requirements validation 4. Requirements management

Q.9 Define functional and non functional requirements. (Refer sections 2.2.1 and 2.2.2)

AU : Dec.-10, May-14

Q.10 Give two examples of non functional requirements. **AU : Dec.-14**

Ans. : Consider the Library Management System for which the two non functional requirements can be -

1. The user who wishes to read the article on-line must be authenticated first.
2. The article must be displayed within 5 seconds.

Q.11 What is the need for feasibility analysis ? **AU : May-15**

Ans. : The feasibility study is required to decide whether or not the proposed system is worthwhile to implement.

Q.12 Define feasibility study and list the types. **AU : Dec.-15**

Ans. : Definition : A feasibility study is a study made to decide whether or not the proposed system is worthwhile.

Types : 1) Technical feasibility 2) Economic feasibility
3) Schedule feasibility 4) Operational feasibility

Q.13 List the characteristics of good SRS. **AU : May-16, Dec.-20**

Ans. : 1) SRS must be **correct.** 2) SRS must be **unambiguous.**
3) SRS must be **complete.** 4) SRS must be **consistent.**
5) SRS must be **traceable.**

Q.14 Classify the following as functional / non functional requirements for banking system.

a) Verifying back balance

b) Withdrawing money from bank

c) Completion of transactions in less than one second.

d) Extending the system by providing more tellers for customers. **AU : Dec.-16**

Ans. : a) Functional requirement b) Functional requirement
c) Non-functional requirement d) Non-functional requirement.

Q.15 What is the purpose of petrinet ? **AU : May-17**

OR Define a petrinet. **AU : Dec.-19**

Ans. : Petrinets are rigorously used to define the system. There are three types of components in petrinets. These are

i) Places : Represent possible states of the system.

ii) Transitions : Causes the change in the states.

iii) Arc : Connects a place with transition or transition with places.

Q.16 Differentiate between normal and exciting requirements. **AU : May-17**

Ans. : Normal requirements : The requirements as per goals and objectives of the system are called normal requirements. For example - Handling mouse and keyboard events for any GUI based system.

Exciting requirements : When certain requirements are satisfied by the software beyond customer's expectations then such requirements are called exciting requirement. For example - In word processing software system if there are some exceptional page layout facilities then it falls in exciting requirements category.

Q.17 Draw a use case diagram for an online shopping which should provide provisions for registering, authenticating the customers and also for online payment through any payment gateway like Paypal. **AU : Dec.-17**

Ans. :

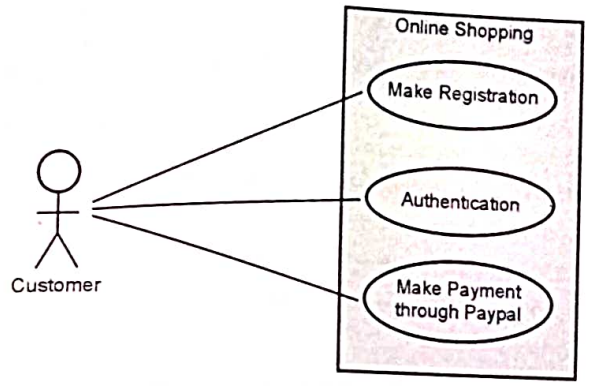


Fig. 2.18.1

Q.18 Define Quality Function Development (QDF). **AU : Dec.-17**

Ans. : Quality function deployment is a quality management technique which translates the customer needs and wants into technical requirements. This technique was introduced in Japan.

Q.19 Draw the context flow graph of a ATM automation system. **AU : Dec.-17**

Ans. :

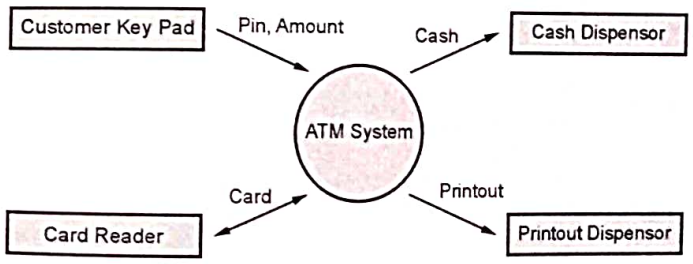


Fig. 2.18.2

Q.20 State two advantages of using Petri Nets **AU : May-19**

Ans. :

1) It specifies the precise definition of problem

2) It represents the graphical workflow.

Q.21 How does data flow diagram help in design of software system ? **AU : May-19**

Ans. : The data flow models are used to model the information and function domain. Refinement of DFD into greater levels help the analyst to perform functional decomposition.

Q.22 Differentiate : Functional and Non-functional requirements. (Refer section 2.2.3)

Q.23 Name any two requirements of elicitation techniques. AU : May-19 AU : Dec.-20

Ans. :

1) **Use cases :** Developing use cases or user stories to describe specific interactions and scenarios involving the software. This technique helps capture functional requirements from a user's perspective.
2) **Questioning :** Continuous asking questions to and seeking clarification from stakeholders during requirements discussions to ensure understanding of the requirements

Q.24 Mention the steps involved in requirements elicitation and analysis. AU : Dec 22

Ans. : The steps are as follows -

1) Requirement discovery
2) Requirements classification and discovery
3) Requirements prioritisation
4) Requirement documentation

❑❑❑