Laboratory Manual

Software Engineering
(CS-403)

For

Second Year Students CSE
Department: Computer Science & Engineering

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Department of Computer Science and Engineering

Vision of CSE Department:

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the CSE Department:

I. To provide high quality education through effective teaching-learning process emphasizing active participation of students.

II. To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.

III. To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Programme Educational Objectives:

Graduates will be able to

I. Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.

II. Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.

III. Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.

IV. Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Program Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Course Outcomes

Software Engineering(CS-403)

CO1 :	Understand basic concepts and identify various models (spiral model, waterfall model concepts) and apply these concepts for designing software application
CO2 :	To design SRS (software requirement specification) for various project. (student management)
CO3 :	Translate a specification to a design, and identify the components to build the architecture for a given problem, using an appropriate software engineering methodology.
CO4 :	To analyze the various testing techniques and apply in specific project(student management)
CO5 :	To analyze the need of Software Maintenance, reengineering and implement these concepts for project Planning.

Course	Course Outcomes	CO Attainment	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	Understand basic concepts and identify various models (spiral model, waterfall model concepts) and apply these concepts for designing software application		**2**			**1**									**1**		
CO2	To design SRS (software requirement specification) for various project. (student management)			**1**												**1**	
CO3	Translate a specification to a design, and identify the components to build the architecture for a given problem, using an appropriate software engineering methodology.		**1**	**1**	**1**												
CO4	To analyze the various testing techniques and apply in specific project(student management)		**1**	**1**							**1**						
CO5	To analyze the need of Software Maintenance, reengineering and implement these concepts for project Planning.			**1**		**1**							**1**			**2**	

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

List of Program

Sr. No.	List	Course Outcomes	Page No.
1	Phases in software development project. (Explain SDLC)	CO1	1-2
2	Analysis report of project i. Abstract Of Project ii. Model Description (Best Suitable for Project) iii. Technology Used iv. References used for project (Similarities, Differences, WebPages)	CO4,CO5	3
3	To perform the system analysis: Requirement analysis, SRS. (According Format)	CO2	4-5
4	To perform the function oriented diagram: DFD or Structured chart.	CO3	6-7
5	To perform UML diagrams using Argo UML tool 5.1 Introduction of UML 5.2 Introduction Argo UML Tool.	CO3	8-9
6	To perform the user's view analysis: Use case diagram. (Based on given Project)	CO3	10-11
7	To draw the structural view diagram: Class diagram, object diagram. (Based on given Project)	CO3	12
8	To draw the behavioral view diagram: Sequence diagram, Collaboration diagram. (Based on given Project)	CO3	13
9	To draw the behavioral view diagram: State-chart diagram, Activity diagram. (Based on given Project)	CO3	14-15
10	To draw the implementation view diagram: Component diagram. (Based on given Project)	CO3	16
11	To draw the environmental view diagram: Deployment diagram. (Based on given Project)	CO3	17
Note: All the designing programs should be designed using Argo UML Tool.			

INSTITUTE OF TECHNOLOGY
& MANAGEMENT
GWALIOR • MP • INDIA

Program 1

Aim: Phases in software development project. (Explain SDLC)

- Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

- SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

SDLC Phases: Given below are the various phases:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

5) Deployment

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

Program 2

Aim: Analysis Report:

A Library Information System

The Oriental Group of Institutes has been recently setup to provide state-of-the-art research facilities in the ield of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application, which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

Program 3:

Aim: To perform the system analysis: Requirement analysis, SRS. (According Format)

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

1.2 PRODUCT SCOPE

1.3 INTENDED AUDIENCE AND DOCUMENT OVERVIEW

1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

1.5 DOCUMENT CONVENTIONS

1.6 REFERENCES AND ACKNOWLEDGMENTS

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

2.2 PRODUCT FUNCTIONALITY

2.3 USERS AND CHARACTERISTICS

2.4 OPERATING ENVIRONMENT

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

2.6 USER DOCUMENTATION

2.7 ASSUMPTIONS AND DEPENDENCIES

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.2 FUNCTIONAL REQUIREMENTS

3.3 BEHAVIOUR REQUIREMENTS

Program 4

Aim: To perform the function oriented diagram: Data Flow Diagram.

Tools/Apparatus Smart Draw

Procedure:

1) Identify various processes, data store, input output etc. of the system and ask students to analyze.
2) Use processes at various levels to draw the DFDs.
3) Identify various modules, input, output etc. of the system and ask students to analyze.

Data flow diagrams:

illustrate how data is processed by a system in terms of inputs and outputs. A Date Flow Diagram (DFD) is a diagrammatic representation of the information (data) flow within a system.

- **Process**: A process transforms incoming data flow into outgoing data flow.
- **Data Store**: Data stores are repositories of data in the system. They are sometimes also referred to as files.
- **Dataflow:** Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
- **External Entities:** External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the systems inputs and outputs.

Description:

Data Flow:

1. Login in data usually contains the ATM card and the PIN number.
2. Accept Info If the Account is verified then the user control can be accept inside the transaction procedure.
3. Reject Info It the account details are not correct, do not grant the permission to access transaction and reject the card.
4. Account Details The details of account are forwarded as the data to be printed in the receipt.
5. Money The Money is given to the user.

Function:

1 Check Account

Input : a. Login Details. Compare the data with the Bank's Database and grant access.
Output : a. Accept Info.
b. Reject

2 Prompts Amount

Input : a. accept info. If the user is valid, then prompt user to enter the amount.
Output : a. account info.

3 Update Database

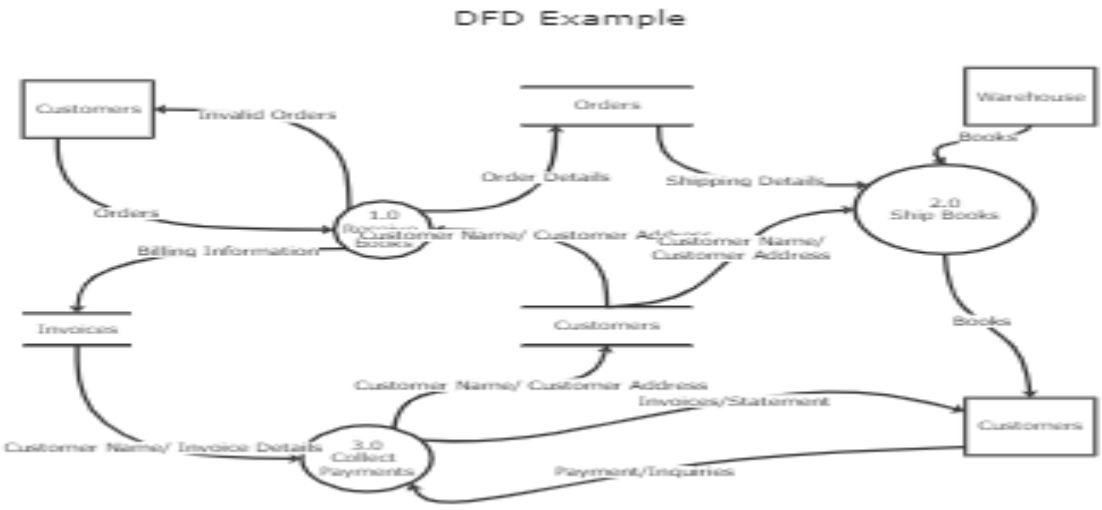
Input : a. account info. When user enters the amount then update the database and perform the transaction.

Output : a. account details.

4. Print Details

Input : a. account details. The details of the changes are printed as the new details of the account.

Output : a. Money.



Program 5:

Aim: To perform UML diagrams using Argo UML tool

5.1 Introduction of UML

5.2 Introduction Argo UML Tool

Q.: What is UML?

A.: "The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other no software systems".— OMG UML Specification

"UML is a graphical notation for modeling various aspects of software systems."

Q.: Why use UML?

A.: Two questions, really:

1) Why use a graphical notation of any sort?
   - Facilitates construction of models that in turn can be used to:
   - Reason about system behavior.
   - Present proposed designs to others.
   - Document key elements of design for future understanding.

2) Which graphical notation should be used?
   - UML has become the de-facto standard for modeling object oriented systems. UML is extensible and method-independent. UML is not perfect, but it's good enough.

The Origins of UML

Object-oriented programming reached the mainstream of programming in the late 1980's and early 1990's. The rise in popularity of object-oriented programming was accompanied by a profusion of object-oriented analysis and design methods, each with its own graphical notation.

Three OOA/D gurus, and their methods, rose to prominence Grady Booch — The Booch Method, James Rumbaugh, et al. — Object Modeling Technique, Ivar Jacobson — Objectory In 1994, Booch and Rumbaugh, then both at Rational, started working on a unification of their methods. A first draft of their Unified Method was released in October 1995. In 1996, (+/-) Jacobson joined Booch and Rumbaugh at Rational; the name UML was coined. In 1997 the Object Management Group (OMG) accepted UML as an open and industry standard visual modeling language for object oriented systems. Current version of UML is 2.0.

UML Diagram Types

There are several types of UML diagrams:

Use-case Diagram

- Shows actors, use-cases, and the relationships between them.

Class Diagram

- Shows relationships between classes and pertinent information about classes themselves.

Object Diagram

- Shows a configuration of objects at an instant in time.

Interaction Diagrams

ArgoUML:

During the 1980's a number of OOA&D process methodologies and notations were developed by different research teams. It became clear there were many common themes and, during the 1990's, a unified approach for OOA&D notation was developed under the auspices of the Object Management Group [http://www.omg.org]. This standard became known as the Unified Modeling Language (UML), and is now the standard language for communicating OO concepts.

ArgoUML was conceived as a tool and environment for use in the analysis and design of object-oriented software systems. In this sense it is similar to many of the commercial CASE tools that are sold as tools for modeling software systems. ArgoUML has a number of very important distinctions from many of these tools.

1. ArgoUML draws on research in cognitive psychology to provide novel features that increase productivity by supporting the cognitive needs of object-oriented software designers and architects.
2. ArgoUML supports open standards extensively—UML, XMI, SVG, OCL and others. In this respect, ArgoUML is still ahead of many commercial tools.
3. ArgoUML is a 100% pure Java application. This allows ArgoUML to run on all platforms for which a reliable port of the Java2 platform is available.
4. ArgoUML is an open source project. The availability of the source ensures that a new generation of software designers and researchers now have a proven framework from which they can drive the development and evolution of CASE tool technologies.

Program 6:

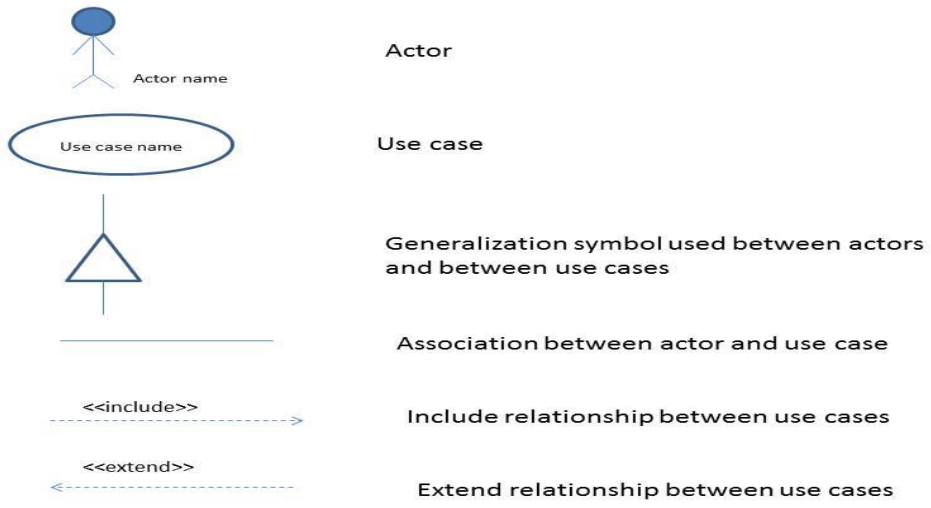
Aim: To perform the user's view analysis: Use case diagram
Tools/Apparatus: Rational rose/ArgoUML

Procedure:

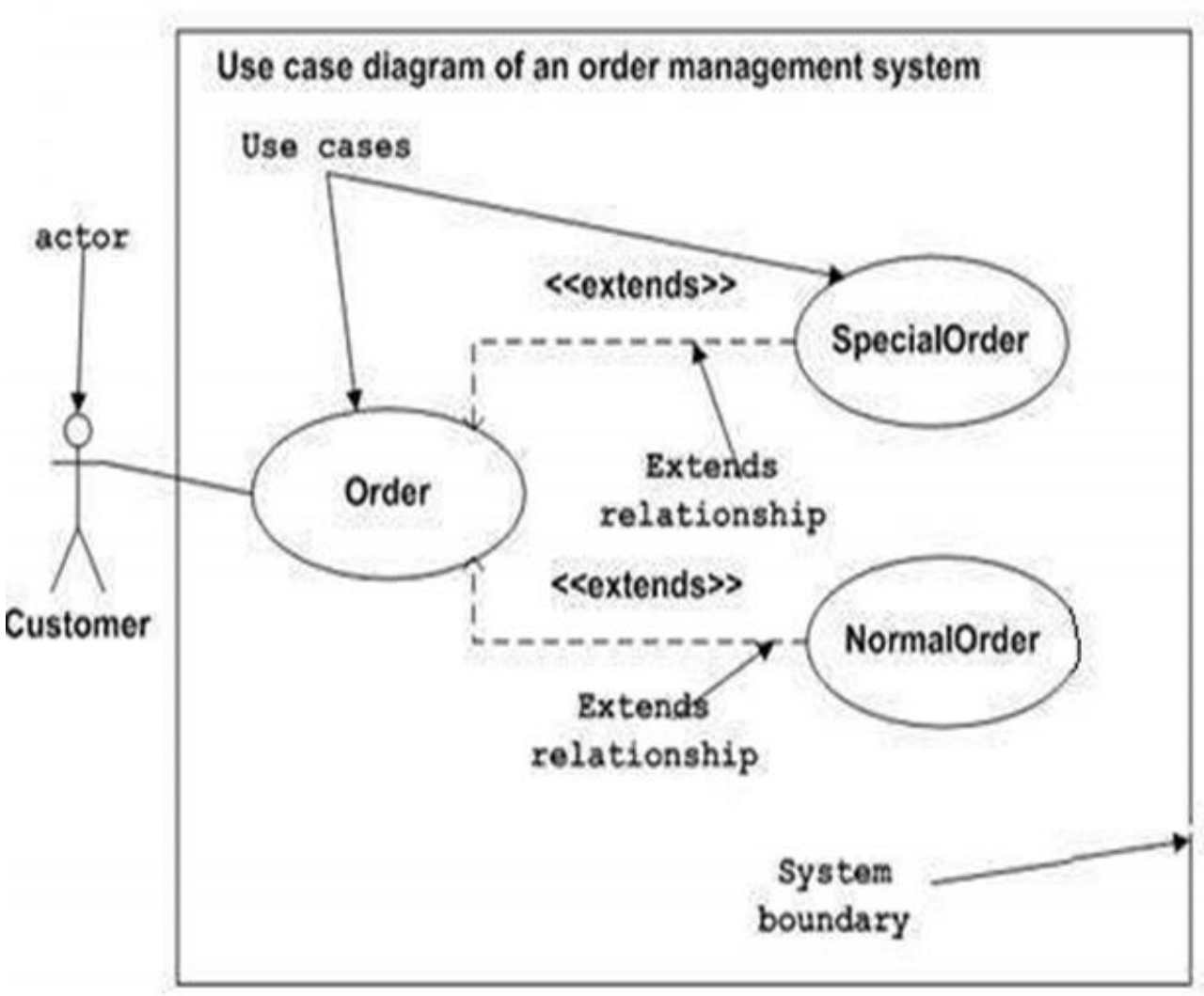
1) Identify various processes, use-cases, actors etc. of the system and ask students to analyze.
2) Use processes at various levels to draw the use-case diagram.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Use Case	Use case identifier and reference number and modification history
Description	Goal to be achieved by use case and sources for requirement
Actors	List of actors involved in use case
Assumptions	Conditions that must be true for use case to terminate successfully
Steps	Interactions between actors and system that are necessary to achieve goal
Variations (optional)	Any variations in the steps of a use case
Non-Functional (optional)	List of non-functional requirements that the use case must meet.
Issues	List of issues that remain to be resolved



Symbols in a use case diagram


Use case diagram of an order management system
Use cases
actor
<<extends>>
SpecialOrder
Order
Extends
relationship
<<extends>>
NormalOrder
Customer
Extends
relationship
System
boundary

Program 7

Aim: To draw the structural view diagram: Class diagram.
Tools/Apparatus:Rational rose/ArgoUML

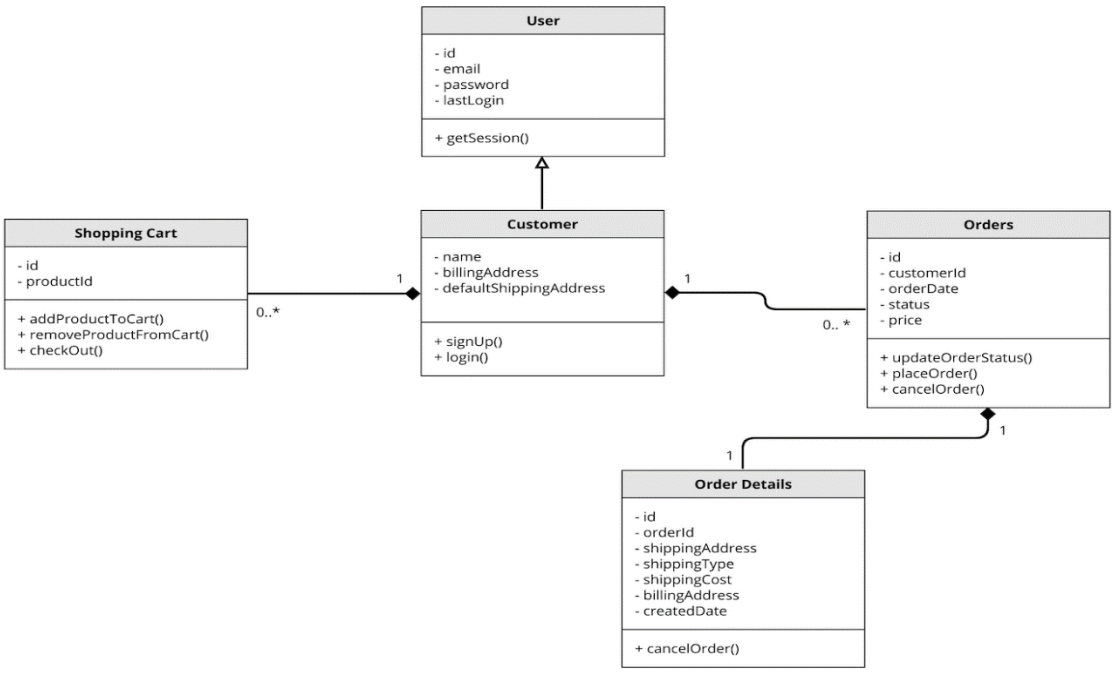
Procedure:

1) Identify various elements such as classes, member variables, member functions etc. of
1) the class diagram
2) Draw the class diagram as per the norms.
3) Identify various elements such as various objects of the object diagram
4) Draw the object diagram as per the norms.

- Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

- Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

- Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.



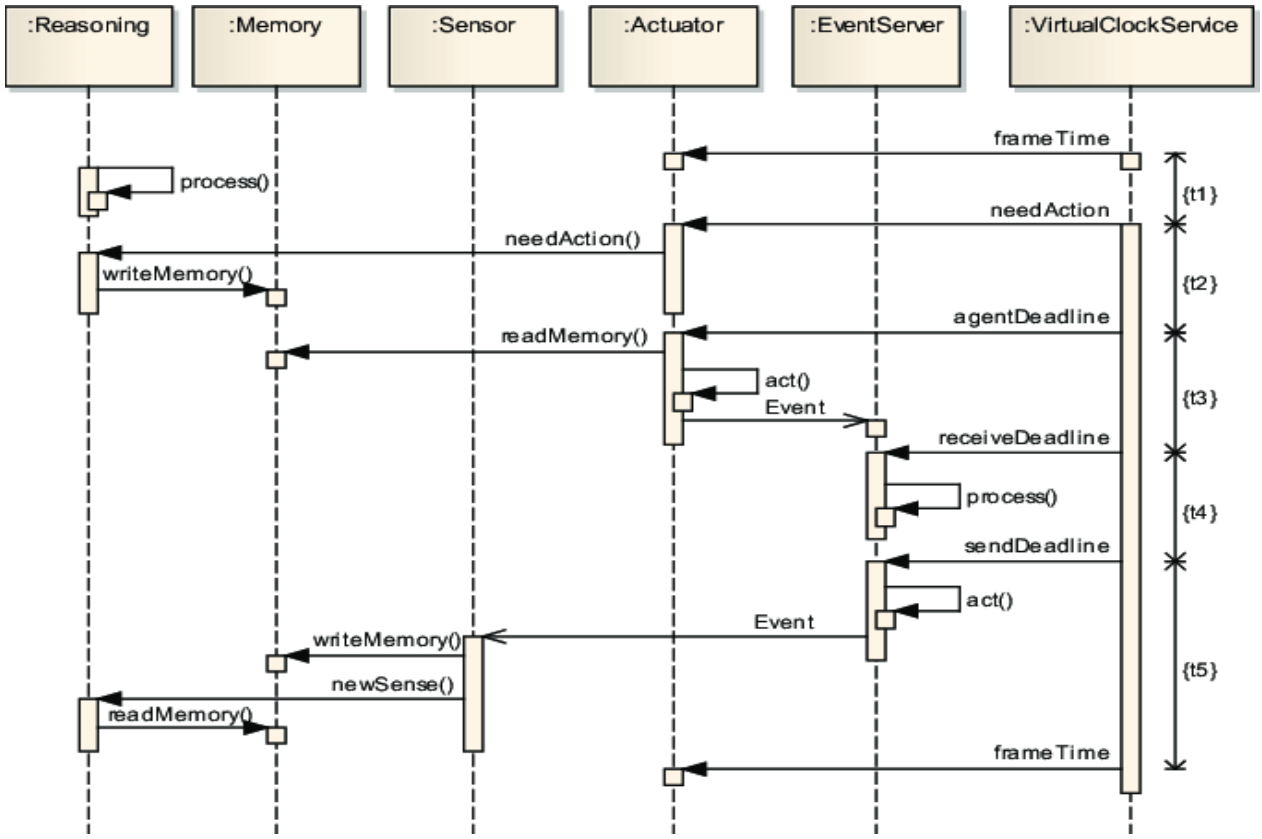
Program 8

Aim: To draw the behavioral view diagram: Sequence diagram.
Tools/Apparatus: Rational rose/Argo UML

Procedure:

1) Identify various elements such as controller class, objects, boundaries, messages etc.
1) of the sequence diagram
2) Draw the sequence diagram as per the norms.
3) Identify various elements such as for the sequence diagram of the collaboration diagram
4) Draw the collaboration diagram as per the norms.

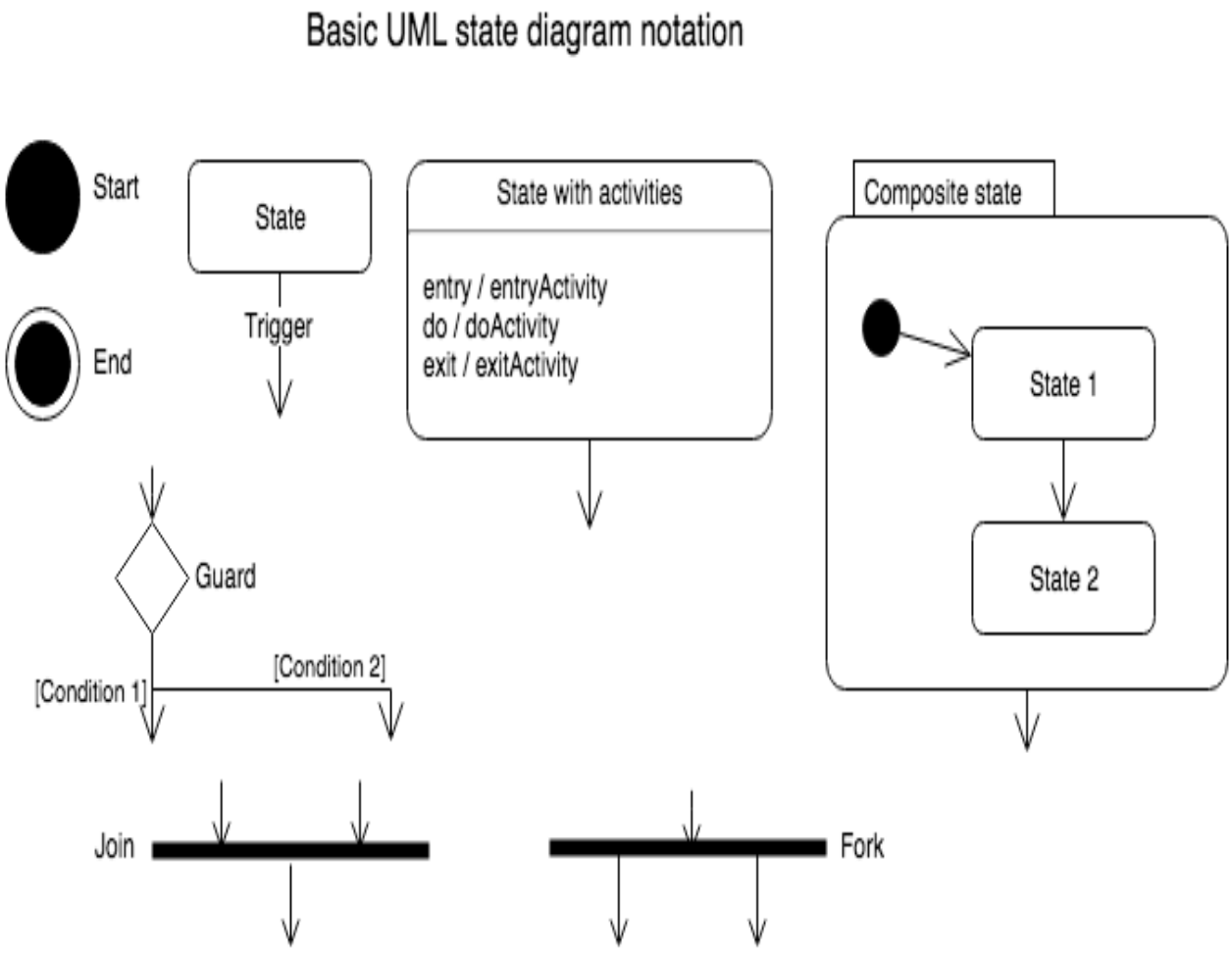
- A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.



Program 9

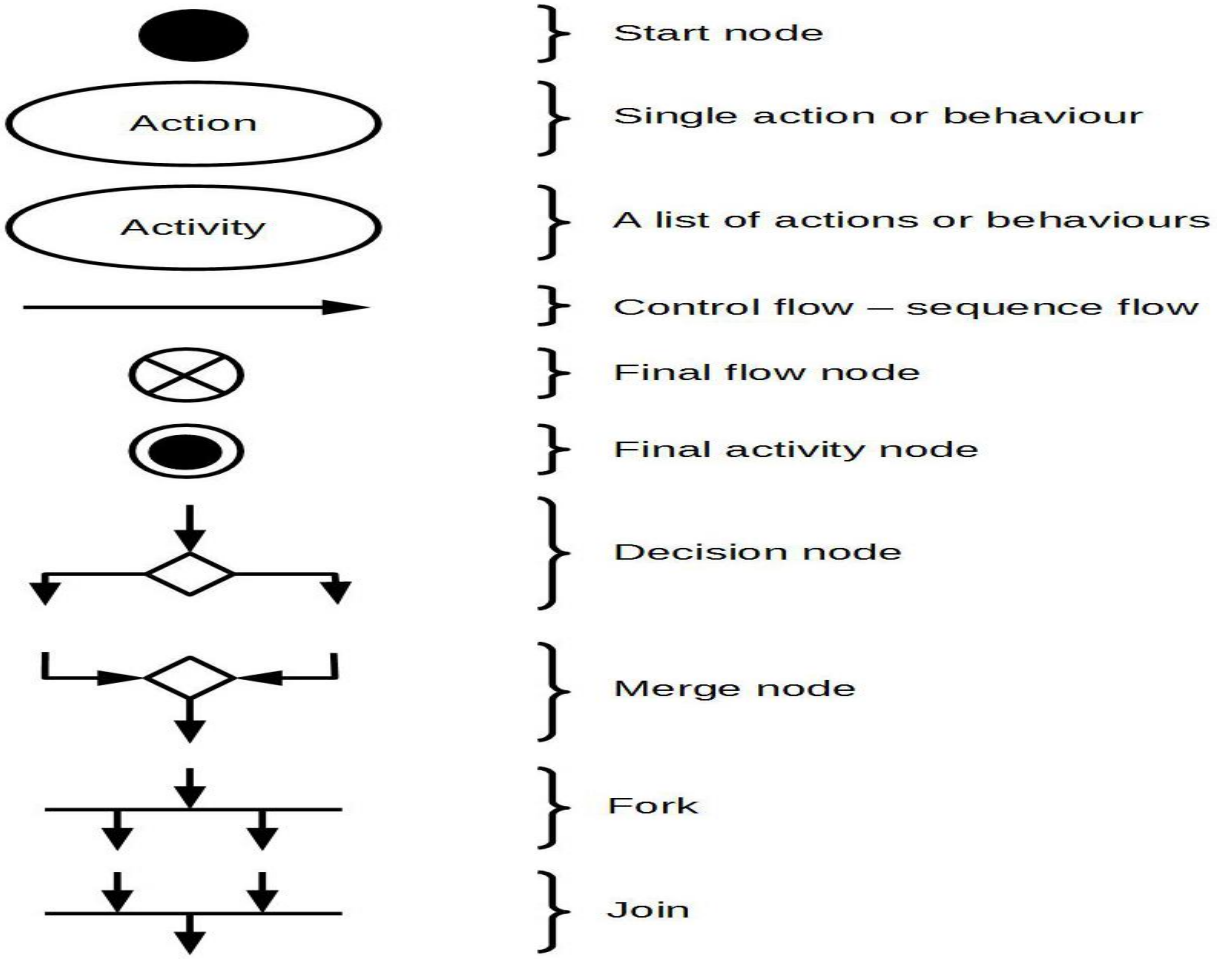
Aim: To draw the behavioral view diagram: State-chart diagram, Activity diagram. (Based on given Project)

State-Chart :- State-chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State-chart diagram is to model lifetime of an object from creation to termination. State-chart diagrams are also used for forward and reverse engineering of a system.



Activity Diagram: The activities that occur within a use case or within an object's behavior typically occur in a sequence is represented by the activity diagram

Basic UML Activity Diagram



Program 10

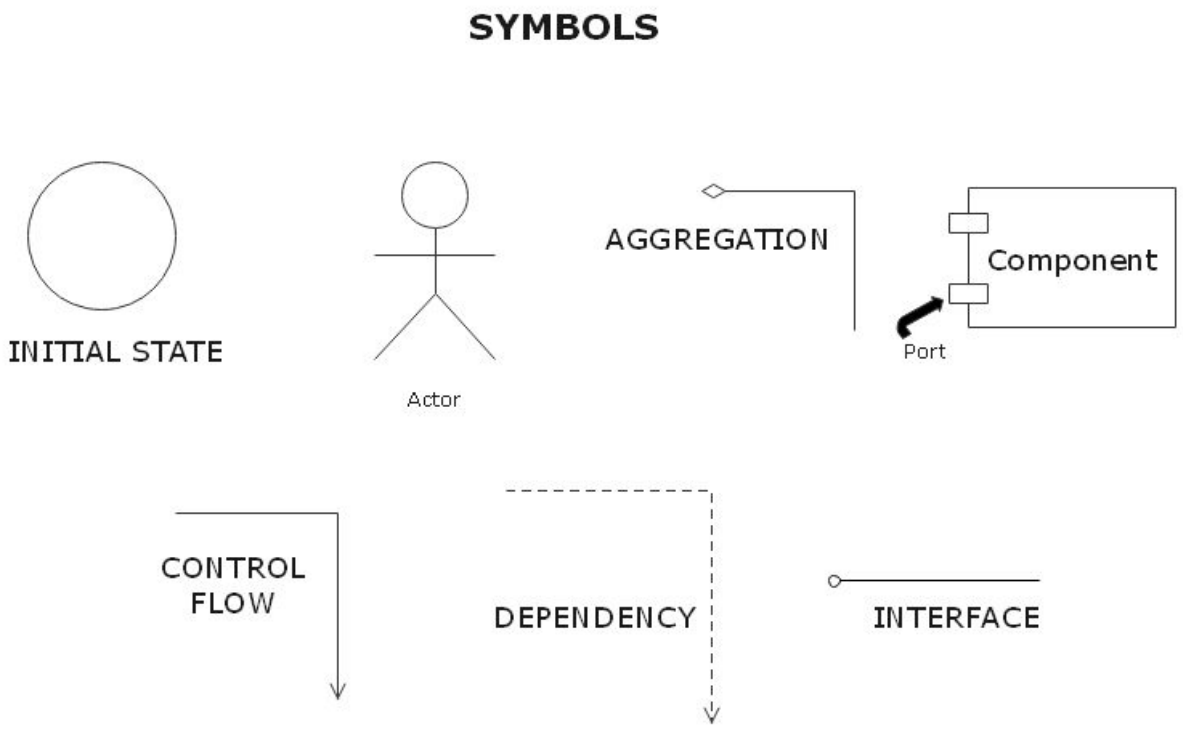
Aim: To draw the implementation view diagram: Component diagram. (Based on given Project)

Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model physical aspects of a system. Now the question is what are these physical aspects?

Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose:
Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. So from that point component diagrams are used to visualize the physical components in a system.

These components are libraries, packages, files etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.



Program 11

Aim: To draw the environmental view diagram: Deployment diagram. (Based on given Project)

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose:
The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed.
Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on software artifacts of a system.
But these two diagrams are special diagrams used to focus on software components and hardware components. So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system.
Deployment diagrams are used by the system engineers. The purpose of deployment diagrams can be described as: Visualize hardware topology of a system.
Describe the hardware components used to deploy software components. Describe runtime processing nodes.