OBJECT ORIENTED ANALYSIS AND DESIGN

BCA - 502

BLOCK 1:
INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED ANALYSIS AND DESIGN

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Prof. K J Sharma

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Mr. Prashant Tikone
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED ANALYSIS AND DESIGN

Contents

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

UNIT 1 INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN

Introduction, Mechanism Of Object-Oriented Approach, Tools And Approaches In OO Analysis And Design

UNIT 2 THE OBJECT PARADIGM

Introduction, Getting Acquainted With The Class Project, Object-Oriented Conceptualization, The Software Life Cycle

UNIT 3 OBJECT ORIENTED METHODOLOGIES

Introduction, Object Oriented Methodologies, Object Process Methodology (OPM)

BLOCK 2: OBJECT MODELLING

UNIT 1 REVIEW OF OBJECT MODELING

Introduction, Principles of Modeling, Object-Oriented Modeling, The Use Case Model

UNIT 2 IMPORTANCE OF MODELING

Introduction, Importance of Modeling, Importance of Business Domain Modelling

BLOCK 3: UML AND OTHER DIAGRAMS

UNIT 1 UNIFIED MODELLING LANGUAGE (UML)

Introduction, Unified Modeling Language (UML) Basics, Practical UML: A Hands-On Introduction For Developers, UML Package Diagram, UML Object Diagram, UML Use Case Diagram, UML Sequence Diagram, UML Collaboration Diagram, UML State chart Diagram, UML Activity Diagram, UML Component Diagram, UML Deployment Diagram, Tips For Effective UML Diagrams

UNIT 2 INTERACTION DIAGRAMS

Introduction, UML Collaboration Diagram, Collaboration Diagrams

UNIT 3 STATE TRANSITION DIAGRAMS

Introduction, State-Transition Diagrams, Domain Expert Testing, Dynamic Modelling - State Transition Diagrams And State charts, State Transition Diagram For A Digital Watch

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

UNIT 1 INTRODUCTION TO COMPONENT TECHNOLOGY

Introduction, Component, Component Characteristics, Components And Objects, Component Extraction From Legacy Software, Software Components – Benefits, Need For Component-Based Development Approach, Different Standards For Component Software, JavaBeans

UNIT 2 CASE STUDY

UML Diagrams Library Management System, Problem Statement, Online Mobile Recharge

Dr. Babasaheb
Ambedkar
Open University

BCA - 502

OBJECT ORIENTED ANALYSIS AND DESIGN

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

UNIT 1
INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN 03

UNIT 2
THE OBJECT PARADIGM 16

UNIT 3
OBJECT ORIENTED METHODOLOGIES 28

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

Block Introduction

Object-oriented technology is very wide and extensive. The users of computer systems found the effects of such technology as increasingly easy-to-use software applications and operating systems which is flexible that is catered by several industries such as banking, telecommunications and television. The major phases of software development using object–oriented methodology are object-oriented analysis, object-oriented design, and object-oriented implementation.

In this block, we will detail about the basic of Object-oriented technology with concepts of various classes. The block will focus on the study and concept of Object-oriented analysis with certain factor related on implementation and limitations in the model. You will get an idea on Software Life Cycle along with its certain features.

In this block, you will make to learn and understand about the basic of Object Process Methodology. The concept related to Object Paradigm and its related features are well explained to you. You will be demonstrated practically about various approaches leading in OO Analysis and Design.

Block Objective

After learning this block, you will be able to understand:

- About Object Oriented Analysis
- Basic of Software Life Cycle
- Features of Object Process Methodology
- Concept of Object Paradigm
- Detailed about Approaches In OO Analysis And Design

Block Structure

Unit 1: Introduction to Object Oriented Analysis and Design

Unit 2: The Object Paradigm

Unit 3: Object Oriented Methodologies

UNIT 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Mechanism of Object-Oriented Approach

1.3 Tools and Approaches in OO Analysis and Design

1.4 Let Us Sum Up

1.5 Answers for Check Your Progress

1.6 Glossary

1.7 Assignment

1.8 Activities

1.9 Case Study

1.10 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of Object-Oriented Approach
- Idea about Object-Oriented Analysis

1.1 Introduction

Object-oriented technology is very wide and extensive. The users of computer systems found the effects of such technology as increasingly easy-to-use software applications and operating systems which is flexible that is catered by several industries such as banking, telecommunications and television. In case of software technology, such object-oriented technology will cover object-oriented management of projects, computer hardware and computer aided software engineering, among others. There are certain concepts that laid the Object-oriented technology to spread its scope which can be:

Concept: Object Behaviour

In earlier programming, we have:

- Data which is inactive
- Functions which will handle any sort of data

It is seen that an object:

- Carries both data and methods which will handle every data
- Active and not passive which does things
- Charges for its own data
- Exploits data to other objects

Concept: Object State

It is seen that in an object state:

- Both data and methods helps to control data
- Data has definite state of object
- Shows relationships among several objects

Concept: Object Classes

It found that in an object class:

- Every object shows a class
- Object has fields or variables
- Field is shown by class
- Object carries methods
- Class represents certainty of suck methods
- Class works as template or cookie cutter

Concept: Classes as Abstract Data Types

In an Abstract Data Type:

- Data is shown by object or thing
- Operations are done on data

Concept: Classes form a hierarchy

In case of classes as hierarchy:

- Classes presents as tree like structure
- Root class is known as object
- Every class excluding object is a superclass
- Carries old object
- To define a class, you need to specify as superclass
- If superclass is not defined, it means object
- Every class has one or more subclasses

Concept: Objects from Super classes

A class can be called as fields or methods

Objects of such class contains fields and methods

Object contains:

- Fields which are in class's super classes
- Methods which are in class's super classes

It is not a detailed for an object.

Concept: Created Objects

It is seen that an n integer will performs:

- Declaration of n integer variable
- Allocation space keeping values for n

Concept: Variables to hold subclass objects

If B is a subclass of A, then:

- A objects will allocate to A variables
- B objects will allocate to B variables

B objects will allocate to A variable, but A objects cannot be allocated to B variable.

All B objects is A but not all A object is B

So bVariable = (B) apbject;

Concept: Constructors make objects

In this:

- Every class has a constructor which generates an object
- Keyword new applies to call constructor as secretary = new Employee ()
- Writing of a constructors if allowed

1.2 Mechanism of Object-Oriented Approach

Object oriented analysis related to the methodology which combines the item that mix with other in terms of class, data or behaviour, thereby creating a model which perfectly shows the required purpose of system. Object-oriented analysis will not factor on the implementation, limitations in the model.

The advantage of using object oriented methodology is to create what you already contain. With such approach, the ability to use code any time and develop additional maintainable systems in less amount of time. Additionally, object-oriented systems are better designed, more resilient to change, and more reliable, since they're built from fully tested and debugged classes.

Check your progress 1

1. Object Oriented analysis includes_____.

   a. Object modeling

   b. Dynamic modeling

   c. Functional modeling.

   d. All of these

1.3 Tools and Approaches in OO Analysis and Design

The major phases of software development using object–oriented methodology are object-oriented analysis, object-oriented design, and object-oriented implementation. Object–Oriented Analysis is a stage where the problem is formulated, user requirements are identified, and model is created on real–world objects. The analysis produces models on how the desired system should function and how it must be developed. The models do not include any implementation

details so that it can be understood and examined by any non–technical application expert. Object–Oriented Design includes two main stages, namely, system design and object design.

There are three main tools used in object-oriented analysis and design techniques:

- Class diagrams templates.
- Object diagrams.
- Object state diagrams

Class diagrams are used to model key abstractions in the problem domain and their relationships with each other. Object diagrams are used to model the interactions between objects, whereas object state diagrams model the dynamic behavior within a single object. An object state diagram shows all the possible states of an object and the allowed transition between the states

Flowchart

A Flowchart is a graphical representation of an algorithm or its process. In this, every step in the process is shown by symbol that gives short description of the process. The symbols in the flow chart are connected together with arrows which show the flow of process in the particular direction. It typically describes the flow of data in a process which shows operations/steps in shape of pictures format that is convenient to understand in textual format.

A flowchart shows the process of operations that works in sequence in order to solve a particular problem. To understand the process, the flowchart gives you the way out step by step. Since it is the pictorial or graphical representation of a process, the idea behind flow chart is to communicate how a process should work. Basically, flowchart analyzes, design and manage a process or program in various fields. Fig 1.1 shows the generic flowchart.

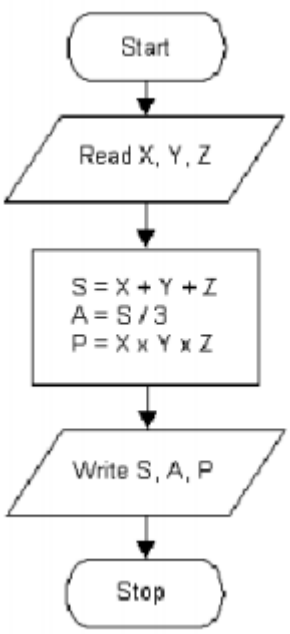


Fig 1.1 Flow chart

It is studied that flowcharts are normally drawn in the beginning of making a computer solution. It serves as a communication tool among the programmers and people. With this, the programming of a particular problem is done that is useful in understanding complicated logic. Once the flowchart is drawn, it becomes easy for programmer to design a program in any high level language.

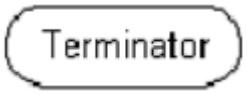
Advantages:

- It is a good way of communicating the logic of a system.
- It is helpful in analysing problems in an effective way.
- It serves as effective program documentation that is required for various jobs.
- It is a kind of blueprint of a particular program which is helpful in development phase.
- It is effective tool for debugging a process.
- It helps the programmer to show efforts efficiently on particular part of program.

Flowchart Symbols:

To draw a flowchart of a particular program or process, you need some symbols to represent it. Some of the standard symbols that are required for drawing flowchart are:

Terminator: It is an oval shape symbol that shows start or end process.

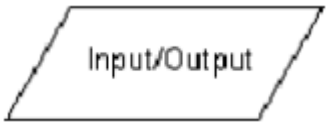


Process: It is a rectangular shaped symbol which shows normal/generic process flow step.

Decision: This is of diamond shaped symbol which shows a branch in the process flow. It is applied when you need to have a decision which is in shape of Yes/No question or True/False.

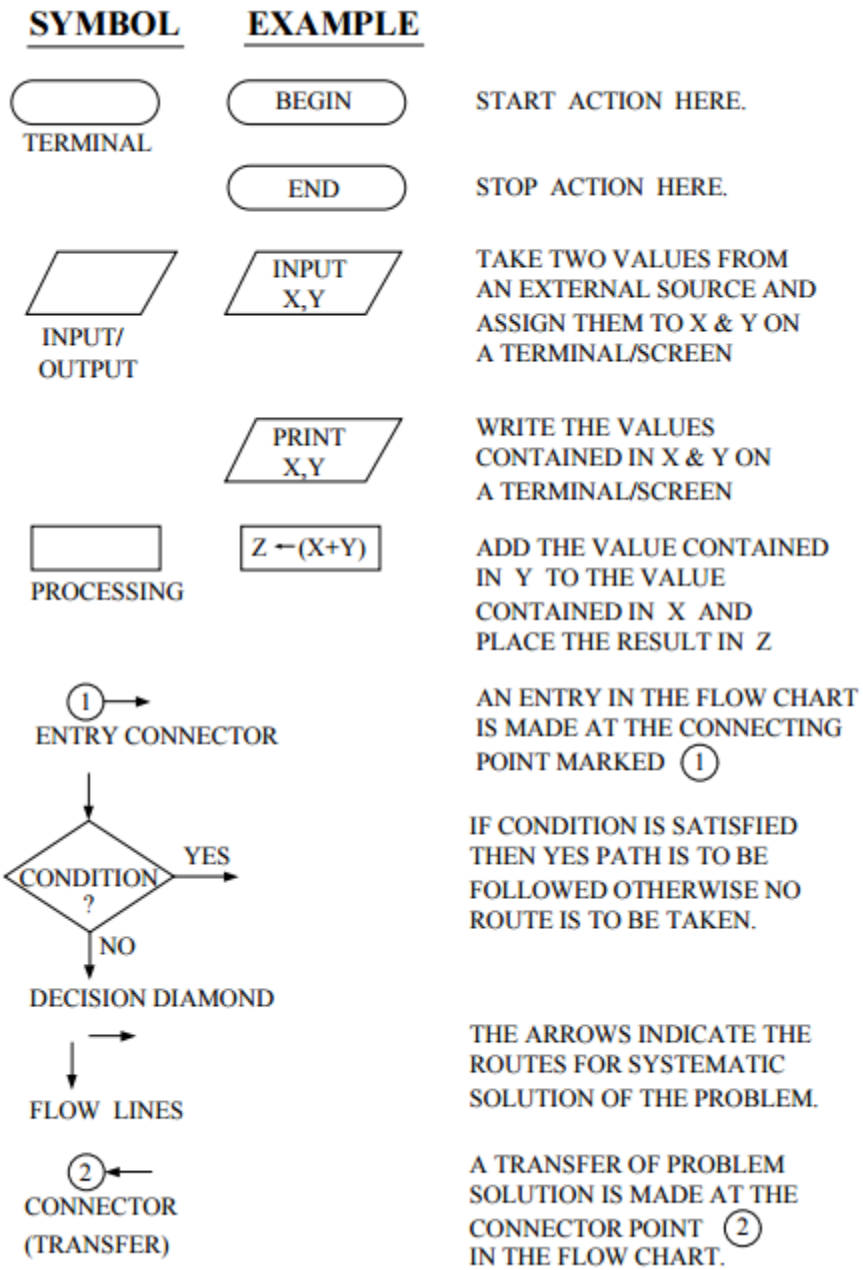
Connector: This is of circular shaped symbol which shows a jump in the process flow.

Data: This is represented as parallelogram shape symbol which shows data input or output (I/O) for a process.



Delay: This is like a D shaped symbol which shows a delay or wait in particular process for input to other process.

Arrow: The arrow shows direction of control flow in a process. Arrow coming from one symbol and entering into another symbol will shows the passing of control to the symbol.



Pseudocode

Pseudocode is a tool which is used for planning program logic. Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

To repeat an instruction, often it is required that all data should be processed in continuation in loop. Loop basically states for as a repetition or iteration of control structure. These codes are made up of following basic logic structures which are used for writing any computer program. These are:

1. Sequence
2. Selection (IF...THEN...ELSE or IF....THEN)

   Iteration (DO...WHILE or REPEAT...UNTIL)

Sequence

It is a logic applied for performing instructions that will work one after another in a particular series. So, in sequence logic, pseudocode instructions are developed in order or sequence where they are used. Logically, the flow of pseudocode is from top to bottom. Figure 1.2 shows an example of sequence logic structure.

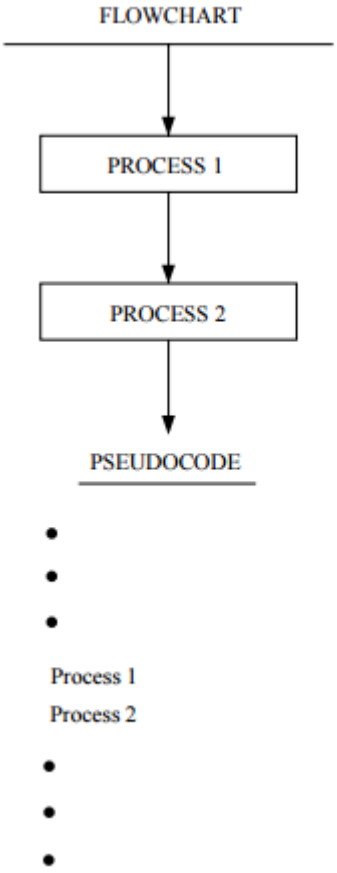


Fig 1.2 pseudocode sequence logic structure

Selection

Such logic is also called as decision logic where the need of decision making is done. This logic will choose the particular path from two or more optional paths located inside the program logic. This logic is either called as IF...THEN...ELSE structural logic or IF.....THEN structural logic. Figures 1.4 and 1.5, shows such type of logical structures with their respective pseudocode.

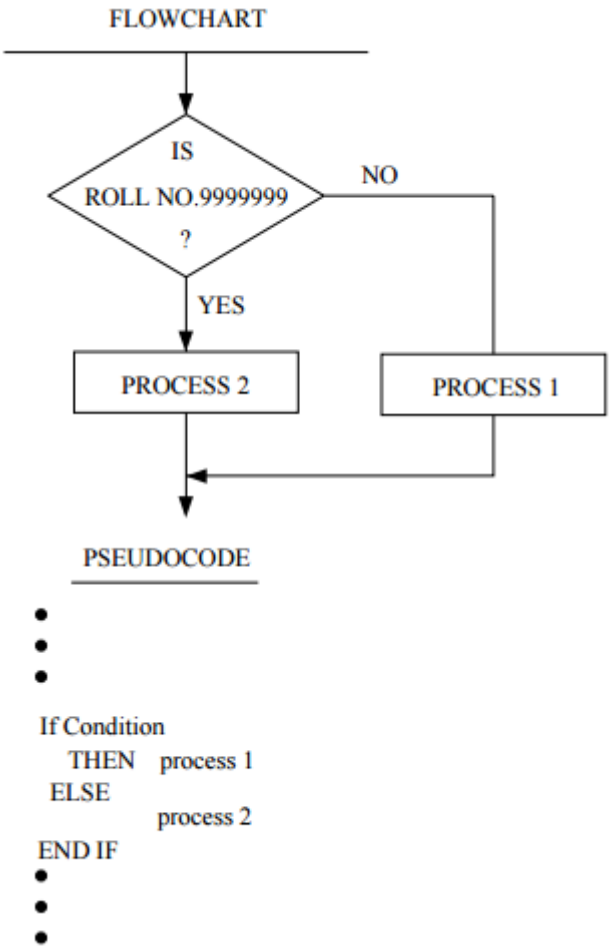


Fig 1.3 pseudocode for If-Then-Else structure

In fig 1.3, the IF...THEN structure shows that when the condition is true, then process 1 will work and if this condition is not true, then the process will skip. As seen, the process 1 and process 2 can be one or more processes.

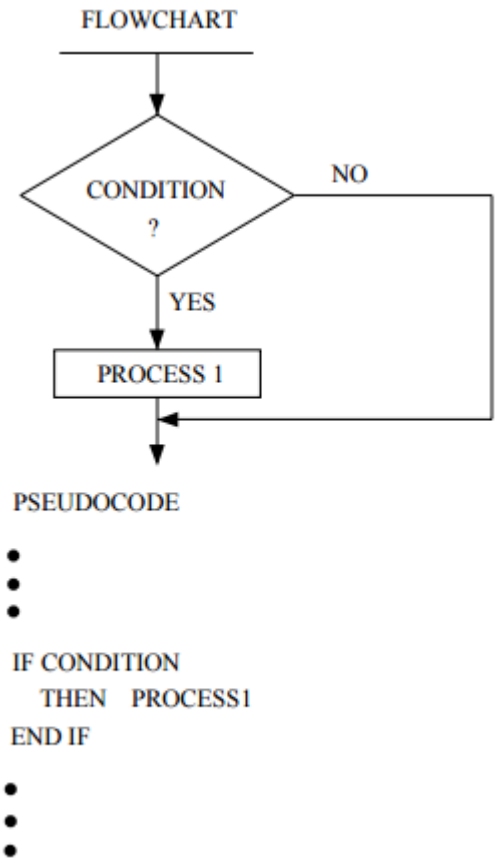


Fig 1.4 pseudocode for If-Then selection structure

Check your progress 2

1. In a flowchart, process can be represented by:

   a. circular shaped object

   b. rectangular shaped object

   c. diamond shaped object

   d. none of above

2. The purpose of Pseudocode is to:

   a. plan for program logic

   b. design logic program

   c. prepare a software

   d. all of above

1.4 Let Us Sum Up

In this unit we have learnt that a programming is a sort of writing instructions sets that guides computer how to do certain work

A Flowchart is a graphical representation of an algorithm or its process in which the diamond shaped symbol represents process flow.

In case of software technology, such object-oriented technology will cover object-oriented management of projects, computer hardware and computer aided software engineering, among others.

1.5 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d)

Check your progress 2

Answers: (1 - b), (2 - a)

1.6 Glossary

1. **Object-oriented analysis -** Process having packed with terms which mix-up with by class, data or behaviour for modeling purpose.
2. **Programming -** It is writing of instructions sets that will guide computer how to do certain work.
3. **Abstraction -** It involves solving of certain program by breaking up into simpler levels.
4. **Pseudocode -** It is a tool which is used for planning program logic.

1.7 Assignment

Explain the Object Oriented approach?

1.8 Activities

Study about features of Object oriented analysis methodology.

1.9 Case Study

Explain the application of tools in OO Analysis and Design.

1.10 Further Readings

1. Norman, Ronald- object oriented system analysis and design –prentice hall 1996
2. Coad.P and Yourdon.E – “Object Oriented Analysis” – Yourdon press

UNIT 2: THE OBJECT PARADIGM

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Getting Acquainted With the Class Project

2.3 Object-Oriented Conceptualization

2.4 The Software Life Cycle

2.5 Let Us Sum Up

2.6 Answers for Check Your Progress

2.7 Glossary

2.8 Assignment

2.9 Activities

2.10 Case Study

2.11 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Concept of Object Paradigm
- Understand about Class Project
- Detailed regarding Object-Oriented Conceptualization
- Basic of Software Life Cycle

2.1 Introduction

Programming is writing of instructions sets which will guide the computer how to do certain work. Doing work relates to reading list of names from a file which could be alphabetical and writing it back again to the file. As seen, it could be much more complex as it involves displaying of graphical user interface which could mainly mean for games and other game's logic. Since it is analysed that Java

is relatively a current object-oriented programming language which has nowadays achieved more popularity and is easy to apply. It is a general-purpose language that can be used for many types of programming tasks.

It is noticed that Object-Oriented Paradigm focuses on real life objects as compared to programming of certain solution. With the real life objects, solutions revolves around various objects that shows respective objects in real life situation.

Benefits of Object-Oriented Paradigm Methodology

Maintainable

Object-Oriented Paradigm makes the code to maintain several times. To locate any source of errors is easy as objects in paradigm are self-contained. The principles of good methods design helps in system maintainability.

Reusable

As there are data and functions in object, objects can be thought of as self-contained boxes which helps in reusing code in new systems.

Scalable

Object-Oriented applications are more scalable then their structured programming roots. As an object's interface provides a roadmap for reusing an object, it also provides you with all the information you need to replace the object without affecting others. This makes it easy to replace old and inefficient code with faster algorithms.

2.2 Getting Acquainted With the Class Project

Creating Informal Scenarios

1. Informal scenarios should be short.
2. Informal scenarios should address one activity.
3. Informal scenarios should specify concrete values.
4. Informal scenarios may address some type of user error
5. Implementation details should be omitted.
6. The state of the system prior to initiation should be described.
7. The next scenario should be indicated.

Sample Informal Scenario

Current System State

The system state here carries player at starting location on game board with different identities, each dealt with six cards since the value of every card is irrelevant to such informal scenario which cannot be considered.

Informal Scenario

Players in this has next move, where player spins the spinner that lands on particular number. Another player has white playing piece where player moves this piece one space to the left, one space toward the top of the game board, two spaces to the right, and finally, one space to the top of the game board. Because of the final position of game piece, the player has no extra options and her turn ends.

Next Scenario

This scenario is repeated with the player to the left of another player, hence the next turn will continue.

Check your progress 1

1. Creating informal scenario in games should have:

   a. short

   b. single activity

   c. definite value

   d. all of above

2.3 Object-Oriented Conceptualization

An object model is typically a logical interface, software or system that's modelled through the utilization of object-oriented techniques. It permits the creation of an architectural software or system model before development or programming. An object model is a component of the object-oriented programming (OOP) lifecycle.

Following are the benefits of using the object model are:

- It helps in quicker development of software.

- It is straightforward to maintain. Suppose a module develops an error, then a programmer will fix that specific module, whereas the other elements of the software are still up and running.
- It supports relatively hassle-free upgrades.
- It enables reuse of objects, designs, and functions.
- It reduces development risks, significantly in integration of advanced systems.

But while object-oriented techniques do facilitate the creation of complex software systems, it is important to remember that OOP is not a panacea. Programming a computer is still one of the most difficult tasks ever undertaken by humans; becoming prominent in programming needs talent, creativity, intelligence, logic, the ability to build and use abstractions, and experience.

Object oriented modelling is entirely a brand new way of thinking about problems. This methodology is all concerning visualizing the items by using models organized around real world ideas. Object oriented models help in understanding issues, communicating with experts from a distance, modelling enterprises, and designing programs and database. In object oriented modelling objects and their characteristics are described. In any system, objects come into existence for playing some role. In the method of defining the roles of objects, some options of object orientation are used.

Class and Objects

A class is a collection of things, or ideas that have the same characteristics. Each of these things or concepts is named as object. Classes define the basic words of the system being modelled. Using a set of classes as the core vocabulary of a software project tends to greatly facilitate understanding and agreement concerning the meanings of terms, and other characteristics of the objects in the system. Classes can serve as the foundation for data modelling. In OOM, the term classes is sometimes the base from that visual modelling tools such as Rational Rose XDE, Visual Paradigm function and design the model of systems.

A class could be a pattern, template, or blueprint for a category of structurally identical items. The items created using the class are known as instances. This is often referred to as the "class or `cookie cutter'" view. As you might guess, the instances are the "cookies."

A class could be a thing that consists of both a pattern and a mechanism for creating items supported that pattern. This is the "class as an `instance factory'"

view; instances are the individual items that are "manufactured" using the class's creation mechanism. A class is the set of all items created using a specific pattern. In another way, the class is that the set of all instances of that pattern.

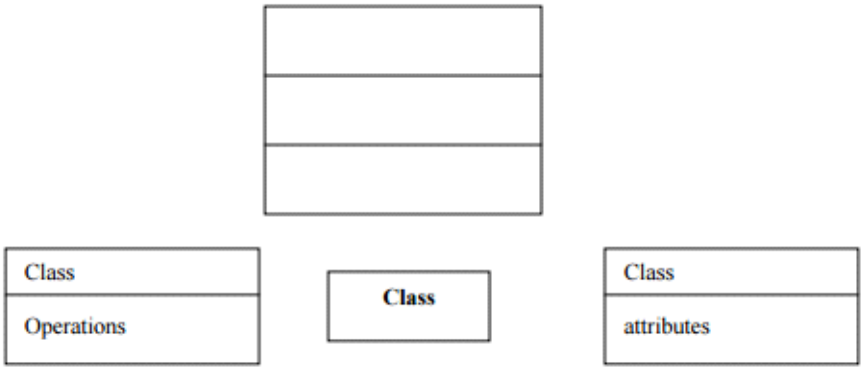


Fig 2.1 Class Notation

Objects

Objects are physical as they are present in universe around us. It is found that hardware, software, documents and concepts are all object samples. For use of modelling, an officer views staff, buildings, divisions, documents and benefits packages as related objects. An automobile person would see tires, doors, engines, speed and fuel level in terms of objects, while atoms, molecules, volumes, and temperatures are objects according to chemist which can be thought of as making an object oriented simulation of chemical reaction.

Normally it is viewed that objects can be considered as particular state. The state of an object is basically the condition of an object, or set of circumstances describing the item. It is generally seen that that people are talking about state information which in particular relates to specific object. It is noticed that state of bank account object will cover latest and available balance, which shows state of clock object that is available at required time showing state of lightweight bulb which could be placed at on or off state. For complex objects like a human being or an automobile, an entire description of the state may be very complex. Fortunately, when we use objects to model real world or imagined things, we have a tendency to generally restrict the possible states of the objects to only people who are relevant to our models**.**

To conceptualize an object-oriented system, once we have selected the objects that comprise the system, we must characterize these objects in terms of their behaviour. Characterizing object behaviour requires thinking about objects of a particular class in relation to objects of other classes. Just as the initial selection of objects reflects a starting point and is expected to change during further analysis,

the initial determination of interobject relationships can be expected to change after further analysis. There are three basic types of relationships that can exist between objects:

- Application specific
- Inheritance
- Aggregation/containment

Check your progress 2

1. Why do there is a need to use object model?
   a. It helps in faster development of software
   b. reusability
   c. reduces development risk
   d. All of these
2. What is class?
   a. Collection of similar objects.
   b. Collection of different types of objects
   c. Group of information
   d. None of these

2.4 The Software Life Cycle

The software life cycle are steps which needs to be followed which includes:

- Design
- Development
- Implement
- Application
- Maintain

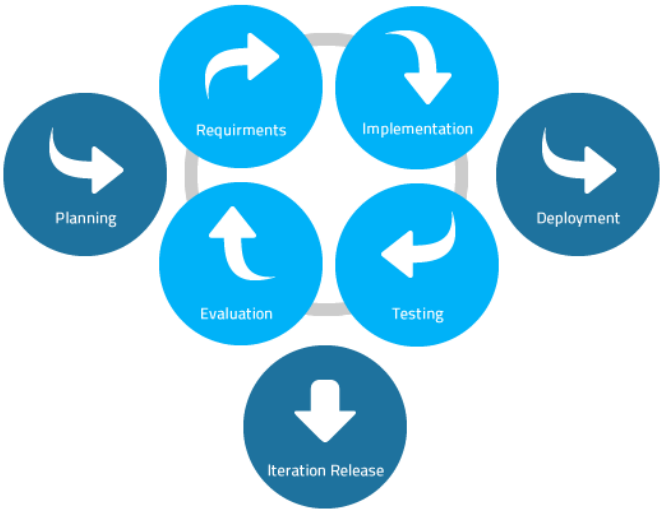


Fig 2.2 Software Life Cycle

Software system, where developer has certain choices to make which decides about type of software development paradigm which need to be applied. Once the development paradigm has been chosen, the developer must choose which software development process to use.

Software project management requires trained and experienced Software Engineers in order to increase the likelihood of project success because software development for large projects is extremely complex and following strict engineering principles will help reduce the risks associated with the project.

Software project management is extremely important as:

- It is difficult to predict as only 10% of projects are delivered in particular budget and on schedule.
- Management result in more effect on success or failure of project than technology advances.
- There exists too much scrap and rework which results in very immature process.

There are various software development models like-

- Waterfall model
- Incremental model
- RAD model
- Prototype model
- Spiral model

A properly managed project has a clear, communicated, and managed set of goals and objectives, whose progress is quantifiable and controlled and whose resources are used effectively to efficiently produce the desired product. When properly managed, a project usually has a communicated set of processes that cover the daily activities of the project, forming the project framework. As a result, every team member understands their roles and responsibilities and how they fit into the big picture, thus promoting the efficient use of resources.

The processes also provide quantifiable feedback (metrics) with respect to process goals and objectives. Metrics provide the information necessary to assess a project's progress against a baseline project plan. Particular attention should be paid to the cost, schedule, scope, and quality aspects of a project. One method of managing these areas is to apply earned value techniques that provide planned verses actual feedback.

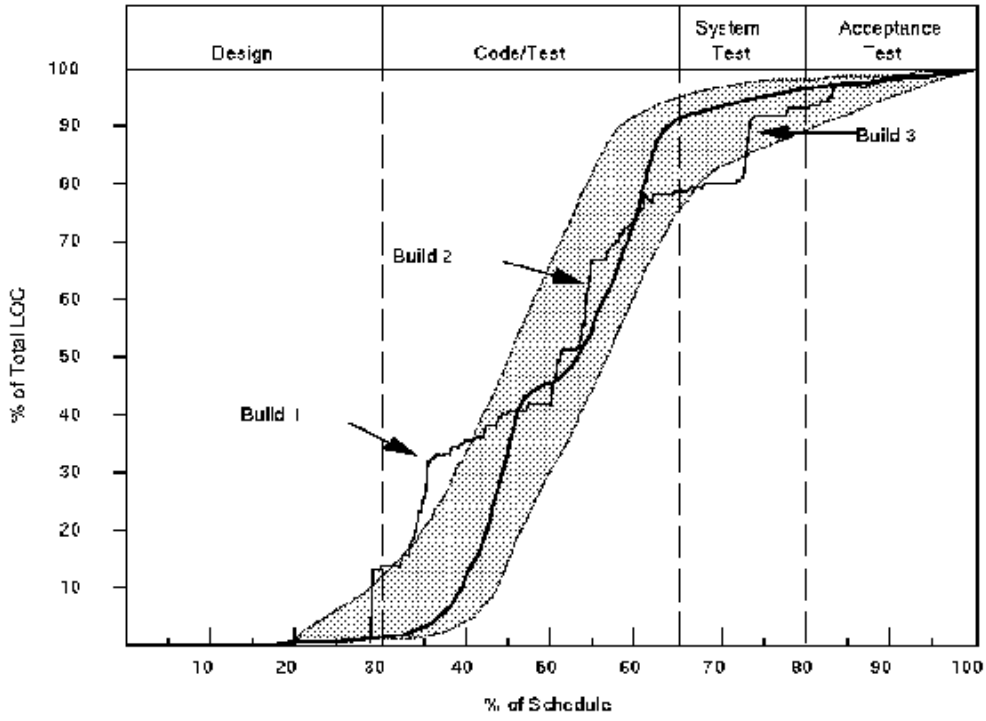


Fig 2.3 Software project Trecking System

Prototyping

Prototyping is the process of building a model of a system. In terms of an information system, prototypes are employed to help system designers build an information system that intuitive and easy to manipulate for end users. Prototyping is an iterative process that is part of the analysis phase of the systems development life cycle.

During the requirements determination portion of the systems analysis phase, system analysts gather information about the organization's current procedures and business processes related the proposed information system. In addition, they study the current information system, if there is one, and conduct user interviews and collect documentation. This helps the analysts develop an initial set of system requirements.

Prototyping comes in many forms - from low tech sketches or paper screens from which users and developers can paste controls and objects, to high tech operational systems using CASE (computer-aided software engineering) or fourth generation languages and everywhere in between. Many organizations use multiple prototyping tools. For example, some will use paper in the initial analysis to facilitate concrete user feedback and then later develop an operational prototype using fourth generation languages, such as Visual Basic, during the design stage.

Advantages of Prototyping:

- Reduces development time.
- Reduces development costs.
- Requires user involvement.
- Developers receive quantifiable user feedback.
- Facilitates system implementation since users know what to expect.
- Results in higher user satisfaction.
- Exposes developers to potential future system enhancements.

Disadvantages of Prototyping

- Can lead to insufficient analysis.
- Users expect the performance of the ultimate system to be the same as the prototype.
- Developers can become too attached to their prototypes
- Can cause systems to be left unfinished and/or implemented before they are ready.
- Sometimes leads to incomplete documentation.

If sophisticated software prototypes (4th GL or CASE Tools) are employed, the time saving benefit of prototyping can be lost.

Check your progress 3

1. What is the goal of Software Project Management?

   a. To build a Software

   b. To plan, measure and control the project development

   c. Maintenance of Software Project

   d. None of the above

2. Prototyping is a _____ process.

   a. Iterative

   b. Linear

   c. Incremental

   d. None of these

2.5 Let Us Sum Up

In this unit we have learnt that object oriented technology is wide and extensive which increases easy-to-use software applications with operating systems that is flexible.

It is seen that object oriented modelling is brand new way of thinking about problems in which you can visualize items by modeling that are arranged around real world ideas.

It is known that class consists of pattern and mechanism for creating items which supports pattern where class as an instance factory can be seen as instances of individual items manufactured by class creation mechanism.

We have studied that links and association are modes of creating link that exists in objects and classes with both links and association bears same feature and establishes link among objects and association establishes class.

The generalization and inheritance are strong abstractions for sharing structure and behaviour of classes and bears relationship among class and show branch of abstraction where subclasses inherit from super classes.

2.6 Answers for Check Your Progress

Check your progress 1

Answers: (1 – d)

Check your progress 2

Answers: (1 - d), (2 – a)

Check your progress 3

Answers: (1 – b), (2 – a)

2.7 Glossary

1. **Object-oriented analysis -** Process having packed with terms which mix-up with by class, data or behaviour for modeling purpose.
2. **Object oriented analysis design -** A bottom up approach which supports system as group of objects that is logical to frame the system.
3. **SPM -** Software project management understand, plan, measure and control of project as delivering on time and budget problems of contiguous and chained allocation.

2.8 Assignment

Write short note on Software Project Management in detail.

2.9 Activities

Collect some information on Software Development Process.

2.10 Case Study

Comment, Incremental model is more flexible than the waterfall model.

2.11 Further Readings

1. W. Shewhart, Statistical Method from the Viewpoint of Quality Control, Dover, 1986.
2. W.E. Deming, Out of the Crisis, SPC Press, 1982; reprinted in paperback by MIT Press, 2003.
3. T. Gilb, Software Metrics, Little, Brown, and Co., 1976.

UNIT 3: OBJECT ORIENTED METHODOLOGIES

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 Object Oriented Methodologies

3.3 Object Process Methodology (OPM)

3.4 Let Us Sum Up

3.5 Answers for Check Your Progress

3.6 Glossary

3.7 Assignment

3.8 Activities

3.9 Case Study

3.10 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of Object
- Concept of Object Modeling
- Idea about Object Process

3.1 Introduction

Object Oriented Methodologies is new system development approach which encourage and facilitate re-use of software components. Such methodologies allow computer system to develop component which allows the effective re-use of current components and facilitate sharing of components by various systems. Using such methodologies will lead to higher productivity, low maintenance cost and good quality.

Object Oriented Methodologies uses international standard Unified Modeling Language (UML) from Object Management Group. UML is a graphical modelling language that supposedly unifies and integrates the various notations used before by the various methodologies proposed. It is the requirement as amount of object-oriented methods increased from less to much higher. It constitutes the required standard notation and semantics for correctly describing software built with object oriented or component-based technology. It's undoubtedly a step in the right direction; however it's not an ideal or universal modelling language. we tend to believe that UML, as it stands today, must be, in some contexts and for a few application domains, complemented with alternative meta-models or a minimum of adapted to address those meta-models.

Object Oriented Methodologies life cycle consists of six stages:

- Business planning stage
- Business architecture definition stage
- Technical architecture definition stage
- Incremental delivery planning stage
- Incremental design and build stage
- Deployment stage

3.2 Object Oriented Methodologies

Object Modeling Technique

The Object Modeling Technique covers analysis, design, and implementation phases of application development.

Extreme Programming (XP)

Extreme Programming (XP) is actually a purposeful and well-organized approach to software development since it stresses customer satisfaction. It is designed to deliver software as required by customer which empowers developers to confidently respond to changing customer needs in life cycle.

It emphasizes team work which involves managers, customers and developers to work as team to deliver quality software which can be performed in simple and effective way to facilitate groupware style development.

Object-Oriented Software Engineering

Object-Oriented Software engineering is a use-case-driven methodology that acts as dialog among user and system. It is a particular form of usage that begins with system initiating some transaction or sequence among users of interrelated events.

It has three phases:

- Analysis: The analysis phase involves examining requirements and robustness and produces a requirements model and an analysis model. The requirements model consists of a use case model, interface description, and a problem domain model.
- Construction: The construction phase includes the design and production processes and results in both a design model and an implementation model.
- Testing: The testing phase covers unit testing, integration testing and system testing and produces a test model.

Responsibility-Driven Design

Responsibility-Driven Design is another active methodology that highlights object behaviour and relationships among other objects. This is dynamic methodology as it finds model objects based on behaviour having two phases:

Exploratory: The exploratory phase involves in finding classes, determining responsibilities, and identifying collaborations using CRC cards.

Analysis: The analysis phase uses refining object's behaviour and service definitions in exploratory phase which includes defining protocols and constructing implementation specifications for every class.

Booch Method

Booch method distinguishes micro and macro elements of development process, where micro development process involves framework for iterative and incremental approach while macro development process involves controlling framework for micro process.

Fusion Method

Fusion method gives systematic approach to object-oriented software development as it carries integrated and extended existing approaches to give direct route from requirements definition using program language implementation.

UML

Unified Modeling Language specifies, visualize, construct and document artifacts of software systems for business modeling and other non-software systems. It was framed with ideas and concepts from OOSE, OMT-2 and Booch methods. OMT-2 was revision of OMT having formal introduction of use cases into methodology. OMT-2 has special expressiveness for analysis and data-intensive info systems and since that special type of systems have sometimes a trivial functional model, it's simple to know the non-inclusion of DFDs among UML.

Visual Modeling Technique

VMT methodically and in a novel way combines techniques that have proven productive in other object methodologies.

Check your progress 1

1. Requirements model uses:

   a. use case model

   b. interface description

   c. problem domain model

   d. all of above

3.3 Object Process Methodology (OPM)

Object-Process Methodology is holistic approach about systems which integrates object-oriented and process oriented paradigms in one frame of reference. The elements of OPM ontology are entities and links which generalized object and process of any system in OPM. Objects are things which occur while processes are things that transform objects. At a particular point in time, object can be in one state and its states changes through occurrences of processes. Analogically, links can be structural or procedural. Structural links shows static relations among pairs of entities, while procedural links connect entities to explain behaviour of a system.

Object Process Methodology is systems development approach which shows challenges which problems with aforementioned OO methods rise. Using a single,

integrated graphic and natural language model, it caters natural train of thought developers to understand and build complex systems with humans, hardware and software. Its structure and behaviour are intertwined so tightly, that on separation results in complication of description. OPM achieves model integration by using three major aspects:

- function
- structure
- behaviour

In single model where objects and processes are shown without suppressing each other. It counters contemporary object oriented systems development methods, which require several models to completely specify a system. OPM is therefore not another OO analysis and design method, as it recognizes fact which separates structure from behaviour involving system modelling resulting into model multiplicity problem which is counter intuitive and productive.

Check your progress 2

1. Model Integration in Object Oriented Methodology is based on:

   a. behaviour

   b. function

   c. structure

   d. all of above

3.4 Let Us Sum Up

In this unit we have learnt that Object-Process methodology is holistic approach about systems which integrates object-oriented and process oriented paradigms in one frame of reference.

The dynamic model describes about the time of performing an operation, while functional model describes about how the operations gets done and which type of arguments are required during the operation.

3.5 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d)

Check your progress 2

Answers: (1 - d)

3.6 Glossary

1. **Object -** It is an instance of a class.
2. **Property -** It is an object characteristic.
3. **Method -** It is an object capability which is a subroutine or function associated with a class.

3.7 Assignment

Explain the Class object?

3.8 Activities

Study more about Object Process in Software.

3.9 Case Study

Study features about Object Process Methodology.

3.10 Further Readings

1. Norman,Ronald- object oriented system analysis and design –prentice hall 1996
2. Coad.P and Yourdon .E – "Object Oriented Analysis" – Yourdon press
3. Coad.P and Yourdon .E – "Object Oriented Design" – Yourdon press

Block Summary

In this block, you have learnt and understand about the basic of file system management and input output memory management. The block focuses on Unified Modeling Language with study about meta models features for reactive systems are discussed. The concept of Object and dynamic model features are well explained.

The block detailed about the basic of Object-Process methodology techniques. The concept related to Unified Modeling Language and Responsibility Driven Design is another active methodology has explained to you. You will be demonstrated practically about Extreme Programming technique.

Block Assignment

Short Answer Questions

1. What is the mechanism involved in Object-Oriented Approach?
2. Explain the function of Object Paradigm?
3. Write note on Software Life Cycle?
4. Write short note on Object Process Methodology?

Long Answer Questions

1. Write short notes on certain tools applied in Object Oriented Design?
2. Write short note on Class Project?
3. Write note on Object-Oriented Conceptualization?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any Other Comments

……

……

……

……

……

……

……

……

……..

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
'Jyotirmay Parisar', Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi,
Ahmedabad-382 481.

OBJECT ORIENTED ANALYSIS AND DESIGN

BCA - 502

BLOCK 2:
OBJECT MODELLING

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED ANALYSIS AND DESIGN

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Prof. K J Sharma

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Mr. Prashant Tikone
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED ANALYSIS AND DESIGN

Contents

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

UNIT 1 INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN

Introduction, Mechanism Of Object-Oriented Approach, Tools And Approaches In OO Analysis And Design

UNIT 2 THE OBJECT PARADIGM

Introduction, Getting Acquainted With The Class Project, Object-Oriented Conceptualization, The Software Life Cycle

UNIT 3 OBJECT ORIENTED METHODOLOGIES

Introduction, Object Oriented Methodologies, Object Process Methodology (OPM)

BLOCK 2: OBJECT MODELLING

UNIT 1 REVIEW OF OBJECT MODELING

Introduction, Principles of Modeling, Object-Oriented Modeling, The Use Case Model

UNIT 2 IMPORTANCE OF MODELING

Introduction, Importance of Modeling, Importance of Business Domain Modelling

---

BLOCK 3: UML AND OTHER DIAGRAMS

UNIT 1 UNIFIED MODELLING LANGUAGE (UML)

Introduction, Unified Modeling Language (UML) Basics, Practical UML: A Hands-On Introduction For Developers, UML Package Diagram, UML Object Diagram, UML Use Case Diagram, UML Sequence Diagram, UML Collaboration Diagram, UML State chart Diagram, UML Activity Diagram, UML Component Diagram, UML Deployment Diagram, Tips For Effective UML Diagrams

UNIT 2 INTERACTION DIAGRAMS

Introduction, UML Collaboration Diagram, Collaboration Diagrams

UNIT 3 STATE TRANSITION DIAGRAMS

Introduction, State-Transition Diagrams, Domain Expert Testing, Dynamic Modelling - State Transition Diagrams And State charts, State Transition Diagram For A Digital Watch

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

UNIT 1 INTRODUCTION TO COMPONENT TECHNOLOGY

Introduction, Component, Component Characteristics, Components And Objects, Component Extraction From Legacy Software, Software Components – Benefits, Need For Component-Based Development Approach, Different Standards For Component Software, JavaBeans

UNIT 2 CASE STUDY

UML Diagrams Library Management System, Problem Statement, Online Mobile Recharge

Dr. Babasaheb
Ambedkar
Open University

BCA - 502

OBJECT ORIENTED ANALYSIS AND DESIGN

BLOCK 2: OBJECT MODELLING

UNIT 1
REVIEW OF OBJECT MODELING 02

UNIT 2
IMPORTANCE OF MODELING 11

BLOCK 2: OBJECT MODELLING

Block Introduction

Object oriented design is totally contrary to both structured methodologies and information engineering. "Both structured design and information engineering use function-oriented decomposition rules, resulting in a structure of procedure-oriented program modules that contain programs, subroutines, and functions with solely procedural code.

In this block, we will detail about the basic of Analysis Techniques such as Object Modelling, Dynamic Modelling and Functional Modelling. The block will focus on the study and concept of Structured Analysis and Object Oriented Analysis. You will show comparison about Structured Analysis and Object Oriented Analysis.

In this block, you will make to learn and understand about the Domain Model supporting as group of logical frame system. The concept related to Object-oriented analysis with groups items interaction of class, data or behaviour also explained to you.

Block Objective

After learning this block, you will be able to understand:

- About Object Modelling
- Basic of Object Modelling
- Features of Object-Oriented Modelling
- Concept of Business Domain Modelling

Block Structure

Unit 1: Review of Object Modelling

Unit 2: Importance of Modelling

UNIT 1: REVIEW OF OBJECT MODELLING

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Principles of Modelling

1.3 Object-Oriented Modelling

1.4 The Use Case Model

1.5 Let Us Sum Up

1.6 Answers for Check Your Progress

1.7 Glossary

1.8 Assignment

1.9 Activities

1.10 Case Study

1.11 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Study about Object Modelling
- Study about principles of Modelling
- Study about Object-Oriented Modelling

1.1 Introduction

Object-oriented technology is incredibly wide and extensive. The users of computer systems found the effects of such technology as increasingly easy-to-use software applications and operating systems that are flexible that's catered by many industries like banking, telecommunications and television. Just in case of software technology, such object-oriented technology can cover object-oriented management of projects, computer hardware and computer aided software engineering, among others.

An interface is mixture of abstract methods. a class implements an interface, and can inherit abstract ways of interface it's not a class, however if we tend to

write it, it's equivalent to writing a class any it's seen that a class describes itself as an attributes and behaviours of an object an interface contains behaviours that a class implements.

1.2 Principles of Modelling

It is one of the most objectives of the software engineering discipline to support the complicated and thus error-prone software development task by providing appropriate sophisticated concepts, languages, techniques, and tools to any or all stakeholders involved. a crucial and nowadays commonly accepted approach within software engineering is that the usage of a software development process model, where above all the software development task is separated into a series of dedicated subtasks. a substantial constituent of such a stepwise approach is the development of a system model. Such a model describes the requirements for the software system to be realized and forms an abstraction.

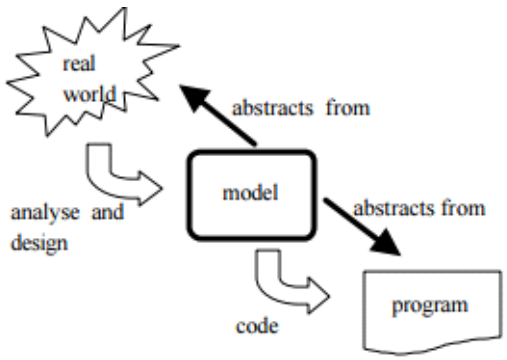


Fig 1.1 Model Representations

The success of object-oriented modelling approaches was hindered within the beginning of the 90ies due to the very fact that surely over 50 object-oriented modeling approaches claimed to be the proper one. This so-called technique war came to an end through an industrial initiative that pushed the development of the meanwhile industrially standardized object-oriented modelling language UML.

Consider an Object Model shown in fig 1.2

Object Modelling

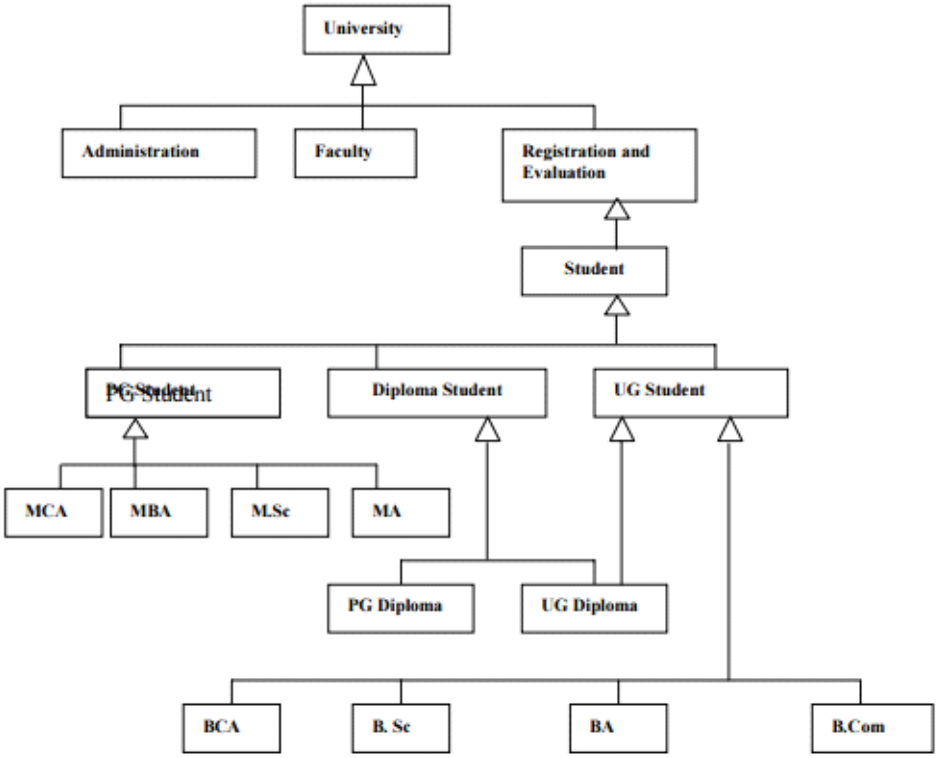


Fig 1.2 Object Model

- An object contains values stored in instance variables within the object.
- Unlike the record-oriented models, these values are themselves objects.
- Thus objects contain objects to an arbitrarily deep level of nesting.
- An object also contains bodies of code that operate on the the object.
- These bodies of code are called methods.
- Objects that contain the same types of values and the same methods are grouped into classes.
- A class may be viewed as a type definition for objects.
- Analogy: the programming language concept of an abstract data type.
- The only way in which one object can access the data of another object is by invoking the method of that other object.
- This is called sending a message to the object.

Check your progress 1

1. What is object oriented modeling strategy?
   a. Use of algorithms
   b. Use of objects
   c. Use of classes
   d. Both B and C
2. What is meant by sending message to object?
   a. Passing parameters to object
   b. Invoking the method of another object
   c. Passing one object definition to another
   d. None of these

1.3 Object-Oriented Modelling

Object Modeling Technique is a methodology that deals with object-oriented development in analysis and design phases. In analysis phase, a problem statement carries list of goals and key ideas within a domain. Problem statement seems in three views or models:

- Object model
- Dynamic model
- Functional model

An object model shows artifacts of system showing static and stable phenomena in model domain with idea related to classes and associations with attributes and operations. Aggregation and generalization are predefined relationships. The notations used in object model are shown in fig 1.3

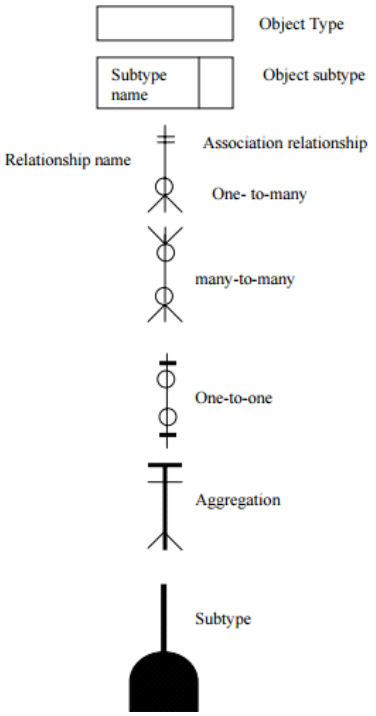


Fig 1.3 Notations in Object Model

The dynamic model shows interaction among such artifacts that shows events, states and transitions. It shows a state/transition view on model having states, transitions that exist among states and events.

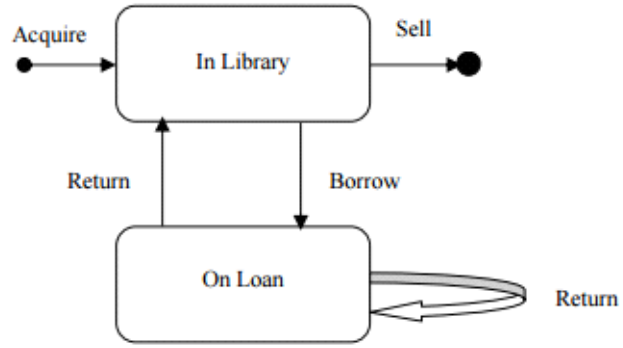


Fig 1.4 State Diagrams

In this, the actions be modelled inside the states. The notations used in this modelling is shown in fig 1.5

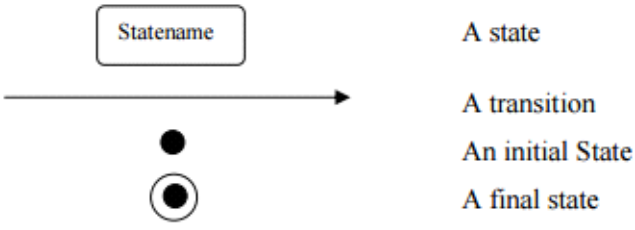


Fig 1.5 Notations

The functional model carries system ways from perspective of data flow. It handles the process viewpoint of model that corresponds to data flow diagrams. It carries main ideas related to process, data store, data flow and actors. Steps employed in framing functional Model comprises of:

- Finding input/output values
- Creating data flow diagrams with functional dependencies
- Explaining the functions
- Locating constraints
- Showing rules of optimization

The system design phase moves along with analysis phase where architecture is established and is arranged in subsystems that allocates processes and tasks with account concurrency and collaboration.

The object design phase is with system design phase wherever plan gets implemented. Object classes are carried out with algorithms having attention to optimize path to persistent data.

Check your progress 2

1. What are benefits of object oriented modeling?
   a. improved reliability
   b. improved flexibility
   c. reduced maintenance
   d. all of these

1.4 The Use Case Model

A use-case model is a model of how different types of users interact with the system to solve a problem. As such, it describes the goals of the users, the interactions between the users and the system, and the required behaviour of the system in satisfying these goals. A use-case model consists of a number of model elements. The most important model elements are: use cases, actors and the relationships between them.

A use-case diagram is used to graphically depict a subset of the model to simplify communications. There will typically be several use-case diagrams associated with a given model, each showing a subset of the model elements

relevant for a particular purpose. The same model element may be shown on several use-case diagrams, but each instance must be consistent. If tools are used to maintain the use-case model, this consistency constraint is automated so that any changes to the model element will be automatically reflected on every use-case diagram that shows that element.

The use-case model may contain packages that are used to structure the model to simplify analysis, communications, navigation, development, maintenance and planning. Much of the use-case model is in fact textual, with the text captured in the use-case specifications that are associated with each use-case model element. These specifications describe the flow of events of the use case.

The use-case model serves as a unifying thread throughout system development. It is used as the primary specification of the functional requirements for the system, as the basis for analysis and design, as an input to iteration planning, as the basis of defining test cases and as the basis for user documentation.

Basic model elements

The use-case model contains, as a minimum, the following basic model elements.

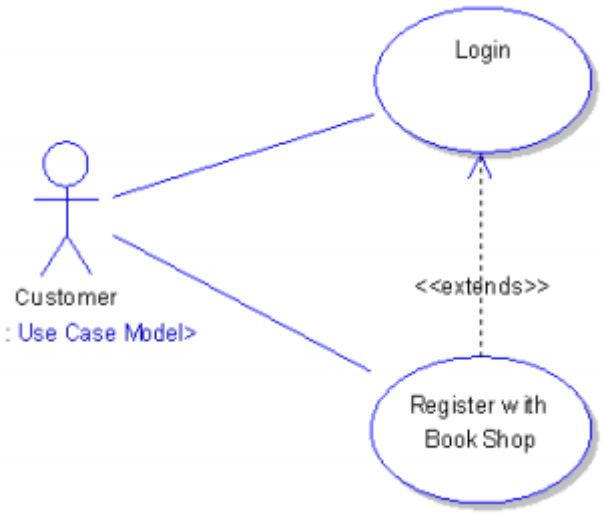


Fig 1.6 Use Case Model

Actor: A model element representing each actor. Properties include the actors name and brief description.

Fig 1.7 Actor

Use Case: A model element representing each use case. Properties include the use case name and use case specification.

Associations: Associations are used to describe the relationships between actors and the use cases they participate in. This relationship is commonly known as a "communicates-association".

Subject: A model element that represents the boundary of the system of interest.

Use-Case Package: A model element used to structure the use case model to simplify analysis, communications, navigation, and planning.

Generalizations: A relationship between actors to support re-use of common properties.

Dependencies: A number of dependency types between use cases are defined in UML. In particular, <<extend>> and <<include>>.

Check your progress 3

1. Which is a hindrance of secondary storage?

   a. slows disk access time

1.5 Let Us Sum Up

In this unit we have learnt that object oriented analysis shows the need of customer which establishes for creating software design, and it defines a collection of needs which will be validated.

It is noted that use-case model carries number of model elements such as cases, actors etc which describes how different types of users interact with system to solve problem.

Dynamic modelling is elaborated further by adding the concept of time: new attributes are computed as a function of attribute changes over time.

1.6 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d), (2 – b)

Check your progress 2

Answers: (1 - d)

Check your progress 3

Answers: (1 – a)

1.7 Glossary

1. **Design -** In modelling, problem related to object domain in programming.
2. **Objects oriented design -** It is the design process which works with one-to-one mapping among problem domain objects and software objects.
3. **Objects design -** It is design strategy where system designers think of operations or functions.
4. **Object-oriented analysis -** Process with terms that carries class, data or behaviour for modelling.

1.8 Assignment

Explain the role of Actor in Use Case Model?

1.9 Activities

Study about Object Modeling Technique.

1.10 Case Study

Study the applications of use-case model.

1.11 Further Readings

1. Norman,Ronald- object oriented system analysis and design –prentice hall 1996
2. Coad.P and Yourdon .E – "Object Oriented Analysis" – Yourdon press
3. Coad.P and Yourdon .E – "Object Oriented Design" – Yourdon press

UNIT 2: IMPORTANCE OF MODELLING

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Importance of Modelling

2.3 Importance of Business Domain Modelling

2.4 Let Us Sum Up

2.5 Answers for Check Your Progress

2.6 Glossary

2.7 Assignment

2.8 Activities

2.9 Case Study

2.10 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Concept of Modeling
- Understand about features of Business Domain Modelling
- Importance of Business Domain Modelling

2.1 Introduction

Object oriented modelling is entirely a brand new way of thinking about problems. This methodology is all concerning visualizing the items by using models organized around real world ideas. Object oriented models help in understanding issues, communicating with experts from a distance, modelling enterprises, and designing programs and database. In object oriented modelling objects and their characteristics are described. In any system, objects come into existence for playing some role. In the method of defining the roles of objects, some options of object orientation are used.

2.2 Importance of Modelling

Here are a number of the advantages of the object-oriented approach:

Reduced Maintenance:

The first goal of object-oriented development is that the assurance that the system can enjoy a longer life while having far smaller maintenance costs. as a result of most of the processes among the system are encapsulated, the behaviours is also reused and incorporated into new behaviours.

Real-World Modelling:

Object-oriented system tend to model the real world in a a lot of complete fashion than do traditional methods. Objects are organized into classes of objects, and objects are related to behaviours. The model relies on objects, rather than on data and processing.

Improved reliability and Flexibility:

Object-oriented system promise to be far more reliable than traditional systems, primarily because new behaviours is "built" from existing objects. because objects is dynamically called and accessed, new objects could also be created at any time. The new objects might inherit data attributes from one, or several other objects. Behaviours are also inherited from super-classes, and novel behaviours is also added without effecting existing systems functions.

High Code Reusability:

Once a new object is created, it'll automatically inherit the data attributes and characteristics of the class from that it absolutely were spawned. The new object will inherit the data and behaviours from all super classes in which it participates. Once a user creates a new type of a widget, the new object behaves "wigitty", while having new behaviours that are defined to the system.

Check your progress 1

1. What are the benefits of object oriented modeling?

   a. improved reliability

   b. improved flexibility

   c. reduced maintenance

   d. all of these

2.3 Importance of Business Domain Modelling

The objects open out by underlying service were bubbled by external service and looks as outside world. By adding domain model to external service, we will able to save external client from excessive versioning. The internal service team need not keep an old version of the service alive, as the external client consumes the external domain model.

Business domain carry out business process which are known as transformation of something from one state to another state by coordination, with the purpose of getting certain goals obtained from responsibility of owner. To help business domain, good information systems software will be applied which will help the organization work through its internal business process and controls every aspects by affecting process. The business process will must be supported with good business process modelling and implementation techniques which analyze, model, and implement business process in professional way to get organizational goals.

There are many companies where we see domains that deals with knowledge of the product, apart from finance such as logistics and processing. Each domains has different needs such as:

Ordering: Requires customer information with all details where not every information is mandatory, that helps in representing customer in many domain models. Warehouse: Here the customer need not to supply any information to pick items, but simply details of product with information about logistics.Shipping: In case of shipping, the customer, gives details about name and address along with contact details.

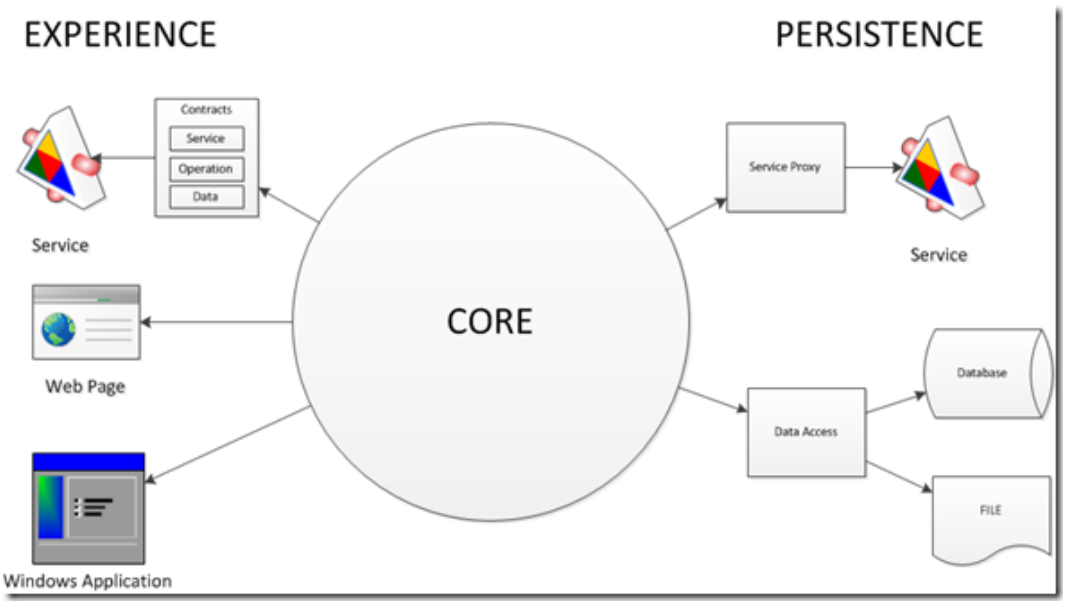


Fig 2.1 Business Domain Model

In the Business Domain Model shown in fig 1.2, the business logic here is application and Persistence and Experience which are not part of application, per se, but extensions so work can get done.

The data tier here is renamed persistence which is Core as Application model where reasoning is application which will not alter based as per persisting data. The only thing that should change when you switch from a file based persistent methodology to a database is your access method.

The business tier has been renamed core, like the Core as Application model. This is due to the fact that this tier contains the application. Switching out the experience layer does not change the application, or does switching to another persistence method.

Domain Driven Modelling

The business domain for any organization adjusts organization business process which explains and modelled for implementation that has business process which defines as transformation of something from one state to another state through partially coordinated agents, with idea of getting goals obtained from responsibility of process owner.

For example defines business processes as ‘‘‘structured sets of activities designed to produce a specified output for a particular customer or market’’. Business processes are similar in different business domains running the same industry of business. To support the business domain, good information systems software is used to support the organization work by handling the internal business process and controlling all aspects affecting the execution of the process.

Check your progress 2

1. Business process relates to:

   a. structured sets of activities designed to produce required output for market

   b. structured sets of activities designed to produce required output for customer

   c. Both a and b

   d. neither a nor b

2.4 Let Us Sum Up

While studying this unit, we have learnt that Object oriented modelling is entirely a brand new way of thinking about problems. This methodology is all concerning visualizing the items by using models organized around real world ideas.

It is seen that business domain carry out business process where it transforms something from one state to another state with the help of coordination in order to have goals from responsibility of owner.

2.5 Answers for Check Your Progress

Check your progress 1

Answers: (1 – d)

Check your progress 2

Answers: (1 - c)

2.6 Glossary

1. **Design -** In modelling, problem related to object domain in programming.
2. **Domain model -** A tool helps in framing conceptual understanding about product or system.
3. **Objects design -** It is design strategy where system designers think of operations or functions.

2.7 Assignment

Write short note on Business domain Modelling.

2.8 Activities

Collect some information on Domain modelling.

2.9 Case Study

Generalised the basic computer architecture and discuss.

2.10 Further Readings

1. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
2. O.M. Nierstrasz, A Survey of Object-Oriented Concepts. In W. Kim, F. Lochovsky (eds.): Object-Oriented Concepts, Databases and Applications, ACM Press and Addison-Wesley, 1989.
3. Wade, S., Salahat, M., Wilson, A Scaffolded Approach to Teaching Information Systems Design, 2012
4. Salahat, M., Wade S. (2014) Teaching Information Systems Development Through an Integrated Framework, Oxford University, Oxford, UK.
5. Williams B. (2005) Soft Systems Methodology, 2010.

Block Summary

In this block, you have learnt and understand about the basic of Principles of Modelling. The block gives an idea on the study and concept of Object-Oriented Modelling along with related features. The students have been well explained on the concepts of Use Case Model.

The block detailed about the basic of importance of Modelling by detailing with object and class. The concept related to importance and necessity of domain models with importance of Business Domain Model are explained to you. You will be demonstrated practically about Domain Driven Modelling.

Block Assignment

Short Answer Questions

1. What is Modeling?
2. Explain the features of Business Domain Model?
3. State the importance of Modeling?
4. Write short note on Business Domain Model?

Long Answer Questions

1. Write short notes on Use Case Model?
2. What are the various principles involved in Modeling?
3. Write note on Object Modelling?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any Other Comments

……

……

……

……

……

……

……

……

……..

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
'Jyotirmay Parisar', Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi,
Ahmedabad-382 481.

OBJECT ORIENTED ANALYSIS AND DESIGN

BCA - 502

BLOCK 3:
UML AND OTHER DIAGRAMS

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED ANALYSIS AND DESIGN

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Prof. K J Sharma

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Mr. Prashant Tikone
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED ANALYSIS AND DESIGN

Contents

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

UNIT 1 INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN

Introduction, Mechanism Of Object-Oriented Approach, Tools And Approaches In OO Analysis And Design

UNIT 2 THE OBJECT PARADIGM

Introduction, Getting Acquainted With The Class Project, Object-Oriented Conceptualization, The Software Life Cycle

UNIT 3 OBJECT ORIENTED METHODOLOGIES

Introduction, Object Oriented Methodologies, Object Process Methodology (OPM)

BLOCK 2: OBJECT MODELLING

UNIT 1 REVIEW OF OBJECT MODELING

Introduction, Principles of Modeling, Object-Oriented Modeling, The Use Case Model

UNIT 2 IMPORTANCE OF MODELING

Introduction, Importance of Modeling, Importance of Business Domain Modelling

BLOCK 3: UML AND OTHER DIAGRAMS

UNIT 1 UNIFIED MODELLING LANGUAGE (UML)

Introduction, Unified Modeling Language (UML) Basics, Practical UML: A Hands-On Introduction For Developers, UML Package Diagram, UML Object Diagram, UML Use Case Diagram, UML Sequence Diagram, UML Collaboration Diagram, UML State chart Diagram, UML Activity Diagram, UML Component Diagram, UML Deployment Diagram, Tips For Effective UML Diagrams

UNIT 2 INTERACTION DIAGRAMS

Introduction, UML Collaboration Diagram, Collaboration Diagrams

UNIT 3 STATE TRANSITION DIAGRAMS

Introduction, State-Transition Diagrams, Domain Expert Testing, Dynamic Modelling - State Transition Diagrams And State charts, State Transition Diagram For A Digital Watch

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

UNIT 1 INTRODUCTION TO COMPONENT TECHNOLOGY

Introduction, Component, Component Characteristics, Components And Objects, Component Extraction From Legacy Software, Software Components – Benefits, Need For Component-Based Development Approach, Different Standards For Component Software, JavaBeans

UNIT 2 CASE STUDY

UML Diagrams Library Management System, Problem Statement, Online Mobile Recharge

Dr. Babasaheb
Ambedkar
Open University

BCA - 502

OBJECT ORIENTED ANALYSIS AND DESIGN

BLOCK 3: UML AND OTHER DIAGRAMS

UNIT 1
UNIFIED MODELLING LANGUAGE (UML) 02

UNIT 2
INTERACTION DIAGRAMS 19

UNIT 3
STATE TRANSITION DIAGRAMS 26

BLOCK 3: UML AND OTHER DIAGRAMS

Block Introduction

Object design is said to finding an object in object oriented analysis phases that are combined into certain classes and are refined so as to get suited for real time implementation. Dynamic modelling is said to numerous components of system having good dynamic behaviour. It shows with State diagrams wherever every state diagram represents single class with important dynamic behaviour whereas it shows interaction among classes.

In this block, we will detail about the basic characteristics of data Flow Diagrams techniques with graphical arrangement of data representing of information. The block will focus on Unified Modelling Language with study about Meta models features for reactive systems are discussed. The concept of Object and dynamic model features are well explained.

In this block, you will make to learn and understand about the basic of Interface template where various functions are implementing with class interface. The concept related to designing of effective UML models along with necessary specifications are well explained to you. You will be demonstrated practically about digital watch layout state diagram and its related technique**.**

Block Objective

After learning this block, you will be able to understand:

- About UML Use Case Diagram
- Basic of UML Collaboration Diagram
- Features of State-Transition Diagrams
- Concept of effective UML Diagrams
- Detailed about Use Case Diagram

Block Structure

Unit 1: Unified Modelling Language

Unit 2: Interaction Diagrams

Unit 3: State Transition Diagrams

UNIT 1: UNIFIED MODELLING LANGUAGE

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Unified Modelling Language (UML) Basics

1.3 Practical UML: A Hands-On Introduction for Developers

1.4 UML Diagrams

1.4.1 UML Package Diagram

1.4.2 UML Object Diagram

1.4.3 UML Use Case Diagram

1.4.4 UML Sequence Diagram

1.4.5 UML Collaboration Diagram

1.4.6 UML State chart Diagram

1.4.7 UML Activity Diagram

1.4.8 UML Component Diagram

1.4.9 UML Deployment Diagram

1.5 Tips for Effective UML Diagrams

1.6 Let Us Sum Up

1.7 Answers for Check Your Progress

1.8 Glossary

1.9 Assignment

1.10 Activities

1.11 Case Study

1.12 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of UML Collaboration Diagram

- Structure about UML Component Diagram
- Types of UML Object Diagram
- Idea about UML State chart Diagram

1.1 Introduction

One of the languages that are gaining exponential popularity and usage is the Unified Modelling Language (UML). It is the requirement as amount of object-oriented methods increased from less to much higher. It constitutes the required standard notation and semantics for correctly describing software built with object oriented or component-based technology. It’s undoubtedly a step in the right direction; however it's not an ideal or universal modelling language. we tend to believe that UML, as it stands today, must be, in some contexts and for a few application domains, complemented with alternative meta-models or a minimum of adapted to address those meta-models.

1.2 Unified Modelling Language (UML) Basics

UML is a graphical modelling language that theoretically unifies and adds many notations which are applied before by various methodologies proposed. It has 2 totally different meta-models:

- state charts
- activity diagrams

These 2 meta-models present several necessary characteristics for reactive systems, particularly concurrency and hierarchy, however they do not allow an elegant treatment of the data path/plant resources and also the specification of dynamic parallelism. These are 2 crucial necessities for embedded software, since totally different parts of the system may attempt to access at the same time the same resources.

UML was created mainly with the ideas and concepts from the OOSE, OMT-2, and Booch methods. OMT-2 was a revision of OMT with the formal introduction of use cases into the methodology. OMT-2 has special expressiveness for analysis and data-intensive info systems and since that special type of systems have sometimes a trivial functional model, it's simple to know the non-inclusion of DFDs among UML.

Check your progress 1

1. UML was initiated by taking idea of:

   a. OOSE

   b. OMT-2

   c. Booch

   d. all of above

1.3 Practical UML: A Hands-On Introduction for Developers

The object is represented in the same way as the class. The only difference is the name which is underlined as shown below.

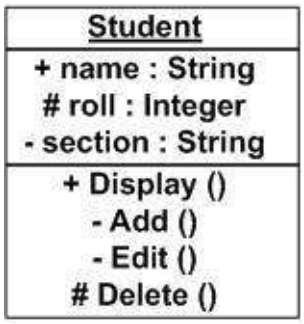


Fig 1.1 Object Notation

As object is actual implementation of a class which is known as the instance of a class. So it has the same usage as that of class.

Interface is represented by a circle as shown below. It has a name which is generally written below the circle.

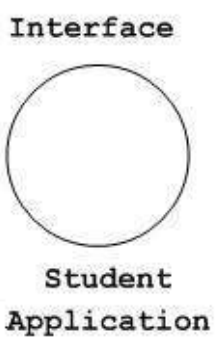


Fig 1.2 Interface Notation

Interface is used to describe the functionality without implementation. Interface is just like a template where you can define different functions and not the implementation. When a class implements the interface it also implements the functionality as per the requirement.

Collaboration is represented by a dotted eclipse as shown below. It has a name written inside the eclipse.

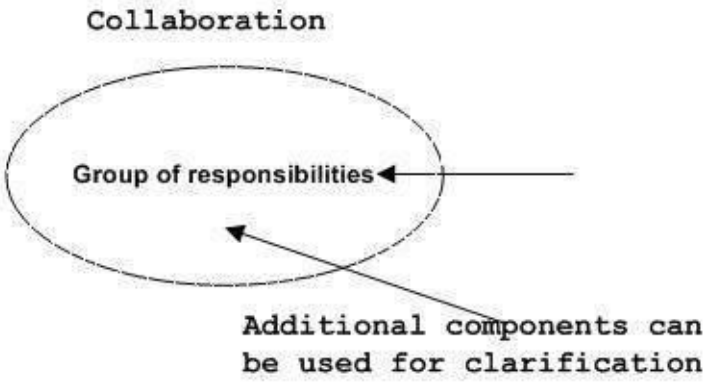


Fig 1.3 Collaboration Notation

Collaboration represents responsibilities. Generally responsibilities are in a group.

Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.

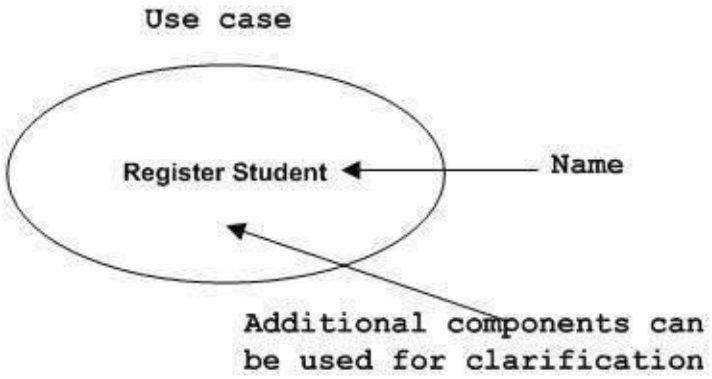


Fig 1.4 Use case Notation

Use case is used to capture high level functionalities of a system.

An actor can be defined as some internal or external entity that interacts with the system.

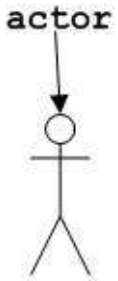


Fig 1.5 Actor Notation

Actor is used in use case diagram to describe the internal or external entities.

Initial state is defined to show the start of a process. This notation is used in almost all diagrams.

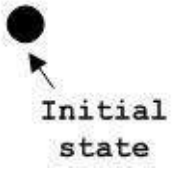


Fig 1.6 Initial state Notation

The usage of Initial State Notation is to show the starting point of a process.

Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.

Fig 1.7 Final state Notation

The usage of Final State Notation is to show the termination point of a process.

Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behaviour of a system.

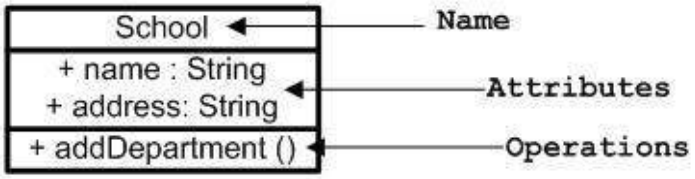


Fig 1.8 Active class Notation

Active class is used to represent concurrency in a system.

A component in UML is shown as below with a name inside. Additional elements can be added wherever required.

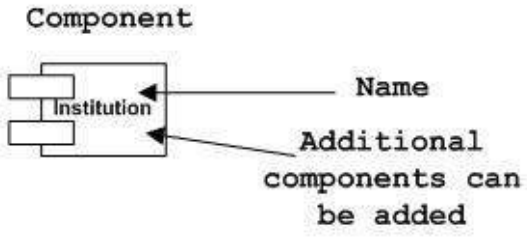


Fig 1.9 Component Notation

Component is used to represent any part of a system for which UML diagrams are made.

A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.

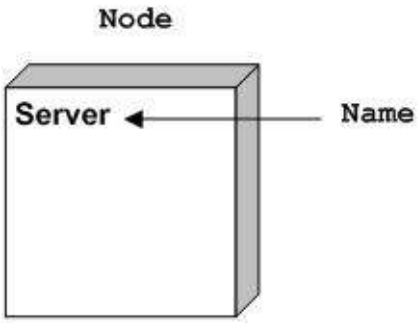


Fig 1.10 Node Notation

Node is used to represent physical part of a system like server, network etc.

Check your progress 2

1. How can we represent an interface?

   a. By using a circle

   b. By using an eclipse

   c. By using a square

   d. None of these

1.4 UML Diagrams

1.4.1 UML Package Diagram

As the name suggests a package diagrams shows the dependencies between different packages in a system.

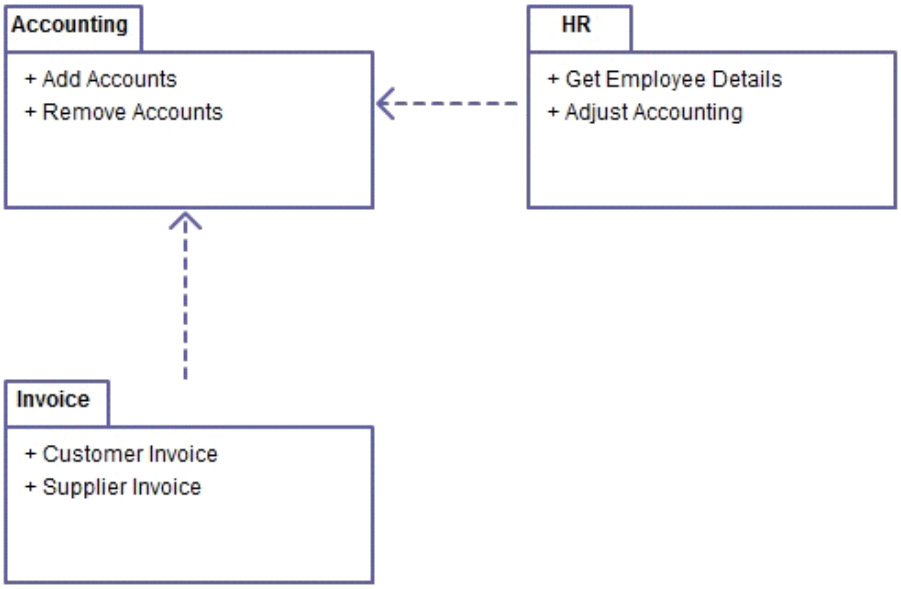


Fig 1.11 Package diagram

1.4.2 UML Object Diagram

Object Diagram is an Instance diagrams which is similar to class diagrams since it describes relationship among objects used in real world. Such diagrams are applied to describe about system appearance at particular time. With support of data available in objects explains complex relationships among objects.

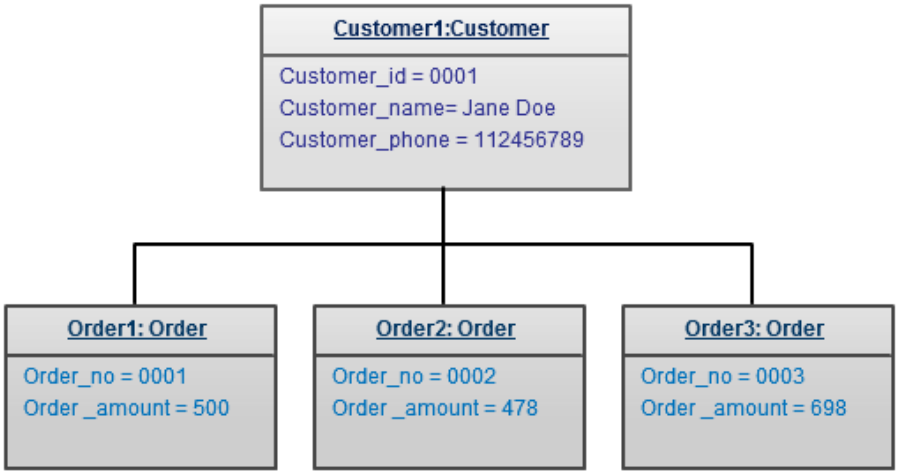


Fig 1.12 Object Diagram

1.4.3 UML Use Case Diagram

The use case model captures the necessities of a system. Use cases are a means of communicating with users and different stakeholders what the system is meant to try to. A use case is a single unit of meaningful work. It provides a high-level view of behaviour observable to somebody or something outside the system. The notation for a use case is an ellipse.

A use case usually includes:

- Name and description
- Requirements
- Constraints
- Scenarios
- Scenario diagrams
- Additional info.

Actors

It is a type of use case diagram which shows interaction with system and entities that is external to system. Such external entities called actors showing roles of human users, external hardware.

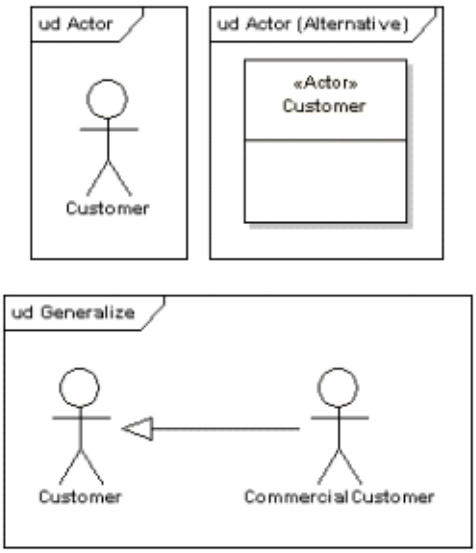


Fig 1.13 Use case diagram

1.4.4 UML Sequence Diagram

Sequence diagrams are structural description about behaviour showing series of sequential steps over time. They represent work flow, message transfer and shows about elements in normal cooperate over time in order to get the result.

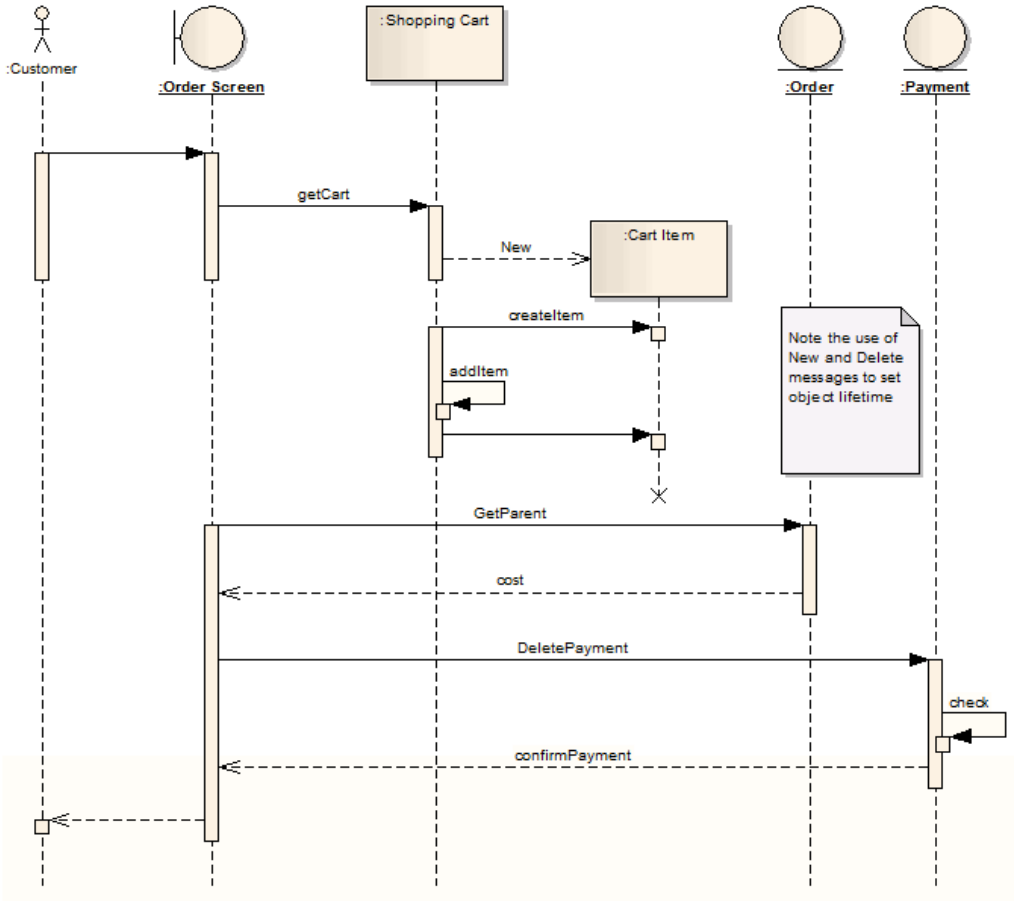


Fig 1.14 Sequence diagram

A Sequence diagram is a structured representation of behaviour as a series of sequential steps over time.

Use to

- Depict work flow, message passing and how elements normally cooperate over time to achieve a result
- Capture the flow of information and responsibility throughout the system, early in analysis; messages between elements eventually become method calls in the class model
- Make instructive models to be used Case scenarios; by creating a Sequence diagram with an Actor and parts involved within the Use Case, you'll model

the sequence of steps the user and therefore the system undertake to complete the required tasks

Construction:

- Each sequence element is arranged in a very horizontal sequence, with messages passing back and forward between elements
- Messages on a Sequence diagram are of many types; the Messages may also be configured to reflect the operations and properties of the source and target elements.
- An Actor part is used to represent the user initiating the flow of events
- Stereotyped elements, like Boundary, control and Entity, is used to illustrate screens, controllers and database things, severally
- Each element has a dashed stem known as a Lifeline, wherever that element exists and potentially takes part in the interactions

1.4.5 UML Collaboration Diagram

It is seen that in an individual Computer, there can be many operations at a particular time; hence the management is required on all running processes that

1.4.6 UML State chart Diagram

State Chart diagrams also known as State Machine diagrams describes about an element that moves among states, which classifies behaviour as per transition, triggers and constraining guards. Naming conventions:

- State Machines were formerly known as State diagrams
- State Machine representations in UML are based on the Harel State Chart Notation and therefore are sometimes referred to as State Charts.

The diagram shown in fig 2.23 represents some of the features of State Diagram.

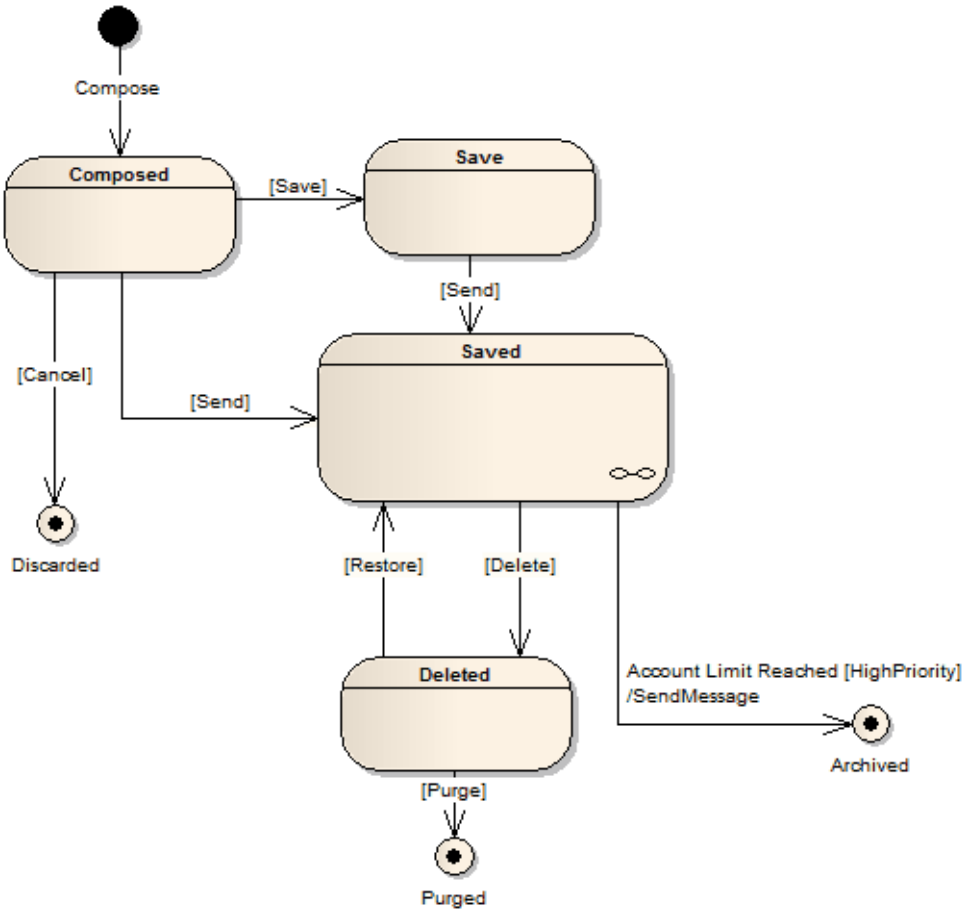


Fig 1.15 State diagram

1.4.7 UML Activity Diagram

Activity diagrams model the behaviours of a system, and also the way in which these behaviours are connected in an overall flow of the system.

Activity diagrams are used to model the behaviours of a system, and also the means during which these behaviours are connected in an overall flow of the system. The logical paths a process follows, supported various conditions, concurrent processing, data access, interruptions and alternative logical path distinctions, are all accustomed construct a process, system or procedure.

The following diagram illustrates some of the features of Activity diagrams, including Activities, Actions, Start Nodes, End Nodes and Decision points.

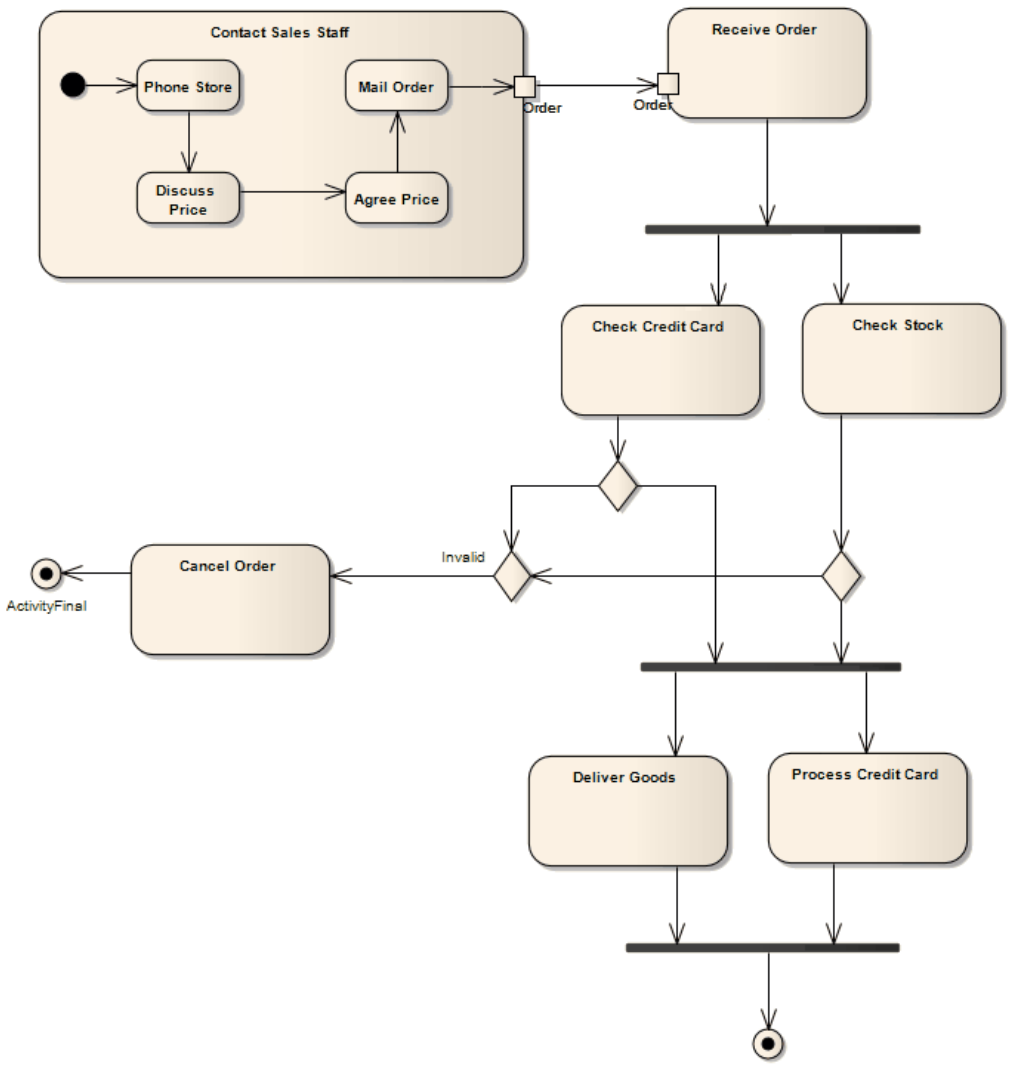


Fig 1.16 Activity Diagram

1.4.8 UML Component Diagram

A component diagram displays the structural relationship of components of the software system. These are generally used when working with complex systems that have many components. Components can communicate with each other using interfaces. The interfaces are linked by using connectors. Below images shows a component diagram.

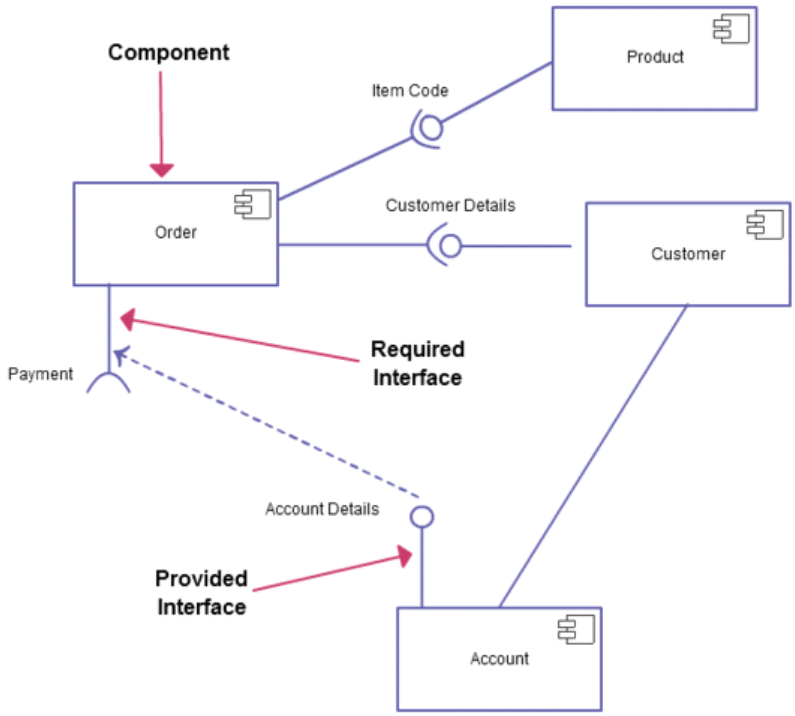


Fig 1.17 Component Diagram

1.4.9 UML Deployment Diagram

A deployment diagram describes various hardware components of system along with software. Such diagrams are required when software solution is put across many machines with each having different configuration as shown in figure:

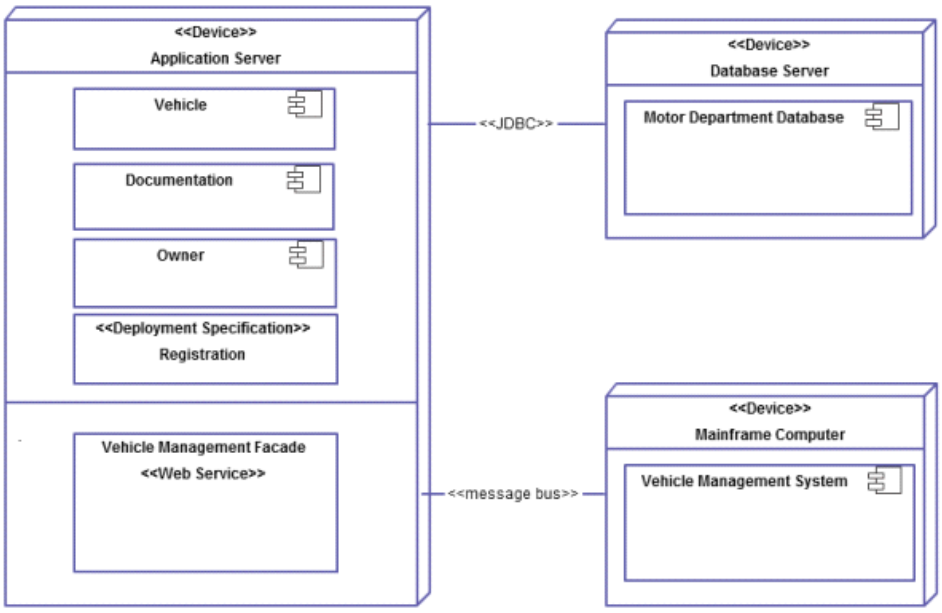


Fig 1.18 Deployment Diagram

Check your progress 3

1. Which of the following is used to describe the movements of elements between states?

   a. State machine diagram

   b. Deployment diagram

   c. Sequence diagram

   d. None of these

2. What are the advantages of using object modelling?

   a. It models the problem domain objects

   b. It can easily mapped the records to corresponding database table

   c. It is used as return values for data Access Object methods

   d. All of these

1.5 Tips for Effective UML Diagrams

UML diagrams are used in modelling of software that helps in designing and analyze software for purpose of development and team management. There are certain tips which need to be followed while drawing good UML diagrams:

The diagram should meet proper abstraction as required as per customer.

There should be no long names and paragraphs in description.

Avoid crossing lines by arranging elements of diagram on page before drawing in the relationship lines. For crossing of line, apply bridge, which is also known as line hop, in order to show intersection of lines?

Apply notes or color to show important features.

Apply UML 2 to drawing UML diagrams as it carries new features that helps in creating nice visual boundary for diagram with proper labelling which is done in namebox which is rectangular in shape with slanted lower right corner.

Initially address the primary structure, behavior and complexities with details. You can link many diagrams using SmartDraw and objects in the chart can be linked to another diagram using Hyperlink.

SmartDraw helps easy sharing of UML diagram with others in case of business presentation by exporting in either image format or PDF.

Check your progress 4

1. In UML 2, the Name Box is:

   a. circular in shape

   b. rectangular in shape

   c. spherical in shape

   d. none of above

1.6 Let Us Sum Up

In this unit we have learnt that UML contains 2 different meta models as state charts and activity diagrams having different characteristics for reactive systems, particularly concurrency and hierarchy.

It is noted that there are important necessities for embedded software as different parts of system attempts to access at same time the same resources.

It is studied that interface describes functionality without implementation which is like a template where different functions are defined without implementation.

A class diagram is an arguably foremost applied UML diagram type which serves as building block of object oriented answer showing classes in exceedingly system, attributes and class operations.

The object diagram is Instance diagrams which is similar to class diagrams that shows relationship among objects used in real world and are applied to describe about system appearance at particular time.

The activity diagrams are used to model the behaviours of a system which means during which these behaviours are connected in an overall flow of the system.

1.7 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d)

Check your progress 2

Answers: (1 - a)

Check your progress 3

Answers: (1 – a), (2 - d)

Check your progress 4

Answers: (1 - b)

1.8 Glossary

1. **UML -** A type of meta models having different characteristics for reactive systems.
2. **Class diagram -** An arguably foremost UML diagram type serving as building block of object oriented answer showing classes in system, attributes and class operations.
3. **Object diagram -** Similar to class diagrams showing relationship among objects in real world that describes about system appearance at particular time.
4. **Activity diagram -** The model showing behaviour of system where behaviours are connected in overall flow of system.

1.9 Assignment

Explain the Tips For Effective UML Diagrams?

1.10 Activities

Study how UML helps developers.

1.11 Case Study

Discuss diagrams involved in UML.

1.12 Further Readings

1. C. Schulte, J. Niere: Thinking in Object Structures: Teaching Modelling in Secondary Schools; in Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts, ECOOP, Malaga, Spain, 2002
2. Zündorf: Rigorous Object Oriented Software Development, Habilitation Thesis, University of Paderborn, 2001.

UNIT 2: INTERACTION DIAGRAMS

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 UML Collaboration Diagram

2.3 Collaboration Diagrams

2.4 Let Us Sum Up

2.5 Answers for Check Your Progress

2.6 Glossary

2.7 Assignment

2.8 Activities

2.9 Case Study

2.10 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Concept of Interaction Diagrams
- Understand about UML Collaboration Diagram
- Detailed regarding Collaboration Diagrams

2.1 Introduction

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

2.2 UML Collaboration Diagram

UML Collaboration diagrams are sort of interaction diagrams that shows the relationship and interaction among software objects. Such diagrams uses cases, system operation contracts along with domain model to already exist. These diagrams shows messages which were sent among classes and objects. Such type of diagram can be created for every system operation which relates to current development cycle.

On creating such diagrams, it is noted that patterns inside it are used to justify relationships which proposed to be best principles for assigning responsibilities to objects that are explained. There are two main types of patterns used for assigning responsibilities such as:

- evaluative patterns
- driving patterns

Every system operation initiates collaboration diagram which means there is collaboration diagram for every system operation. Consider an example diagram for purchasing of bus ticket.

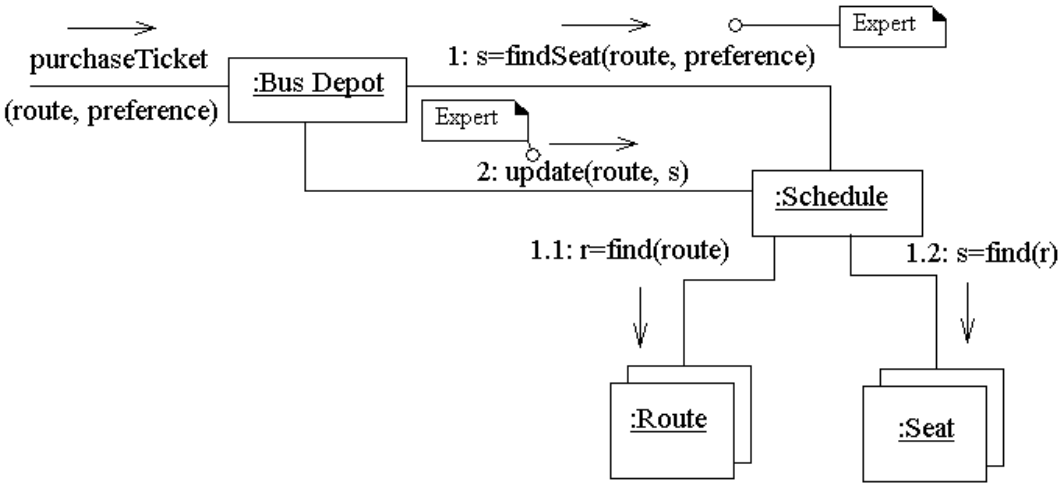


Fig 2.1 Bus Ticket Collaboration Diagram

In the collaboration diagram shown in fig 2.1, the route and seat objects are multi objects that are collection of objects. In this, the message, "purchaseTicket is initialized by initializing actor. Here all other messages are generated by system between objects. The first message after initializing message is numbered. In this, the messages which depends on earlier messages are numbered as per the messages dependencies, hence message, "r=findRoute(route)" is numbered "1.1" as it depends on message "s=findSeat(route, preference)". Any message path that is mutually exclusive is numbered with an "a" or "b". In finding route and seat

messages, if finding a route or a seat were mutually exclusive, the numbering would be 1.1a and 1.1b. Patterns used for the association are associated with the message using a note.

Check your progress 1

1. In UML Collaboration Diagram, patterns are used to:

   a. make objects

   b. justify relationships

   c. arrange numbers

   d. all of above

2.3 Collaboration Diagrams

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams. Collaboration is represented by a dotted eclipse as shown below. It has a name written inside the eclipse.

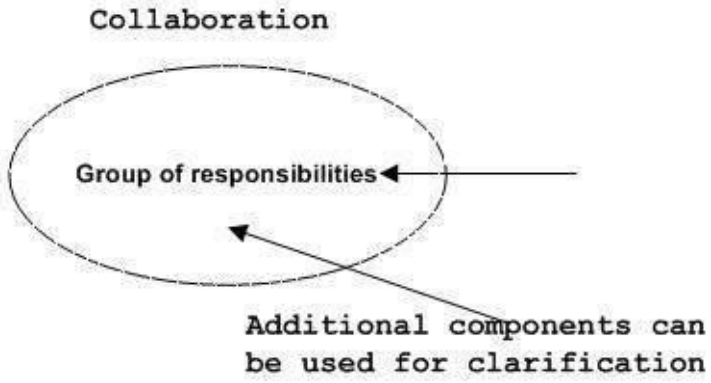


Fig 2.2 Collaboration Notation

Collaboration represents responsibilities which are in group. Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.

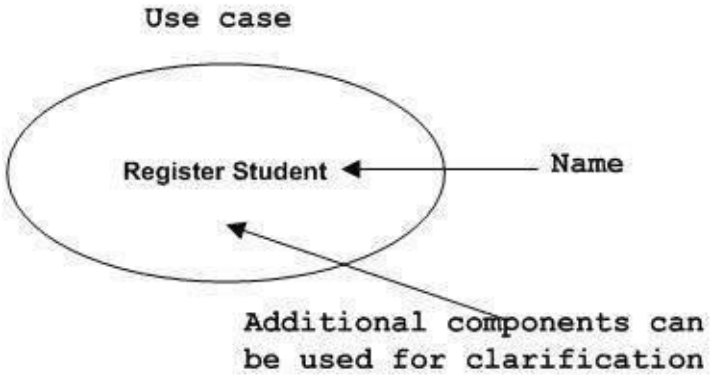


Fig 2.3 Use case Notation

A distinguishing feature of a Collaboration diagram is that it shows the objects and their association with other objects in the system apart from how they interact with each other. The association between objects is not represented in a Sequence diagram. A Collaboration diagram is easily represented by modeling objects in a system and representing the associations between the objects as links. The interaction between the objects is denoted by arrows. To identify the sequence of invocation of these objects, a number is placed next to each of these arrows. The figure shows the representation of collaboration diagram where representation of work flow with message transfer detailed along with elements in normal cooperate over time to get result.

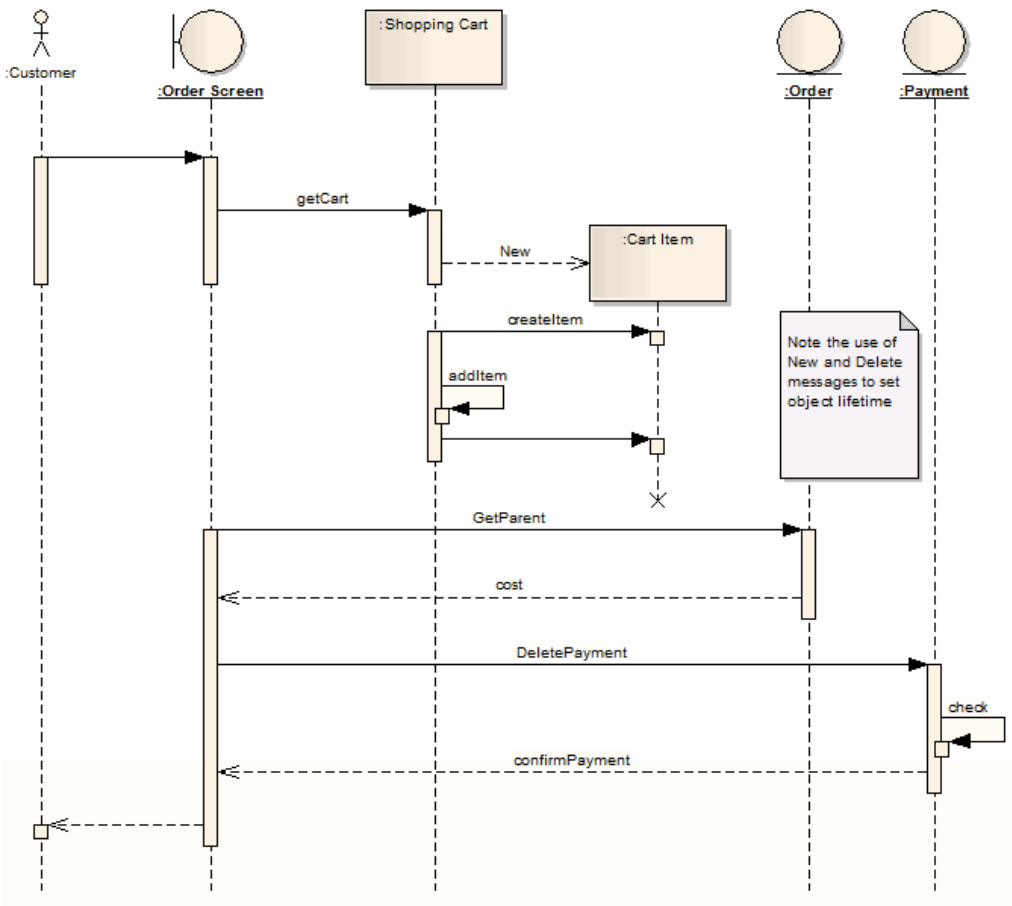


Fig 2.4 Collaboration diagram

A Collaboration diagram similar to sequence diagram is a structured representation of behaviour as a series of sequential steps over time. It is applied to depict work flow, message passing and how elements normally cooperate over time to achieve a result. It helps in capturing flow of information and responsibility throughout the system, early in analysis; messages between elements eventually become method calls in the class model.

Check your progress 2

1. A collaboration diagram is similar to:

   a. flowchart

   b. network

   c. both a and b

   d. none of above

2. In UML collaboration diagram, Use case is shown as:

a. rectangle

b. circle

c. eclipse

d. none of these

2.4 Let Us Sum Up

While studying this unit, we have learnt that collaboration diagram is a type of communication diagram showing relationships and interactions among software objects in Unified Modelling Language.

Collaboration diagrams on the other hand are favourable to portray a simple interaction which occurs among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read

2.5 Answers for Check Your Progress

Check your progress 1

Answers: (1 – b)

Check your progress 2

Answers: (1 - a), (2 – c)

2.6 Glossary

1. **UML -** A type of meta models having different characteristics for reactive systems.
2. **Class diagram -** An arguably foremost UML diagram which serves as building block of object oriented showing classes in system, attributes and class operations.

3. **Object diagram -** A class diagrams showing relationship among objects in real world describing particular system appearance at a time.

2.7 Assignment

Write short note on UML Collaboration Diagram.

2.8 Activities

Collect some information on Collaboration Diagrams.

2.9 Case Study

Generalised about UML Collaboration Diagrams.

2.10 Further Readings

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne
2. Programming Be Operating System by Dan Sydow

UNIT 3: STATE TRANSITION DIAGRAMS

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 State-Transition Diagrams

3.3 Domain Expert Testing

3.4 Dynamic Modelling - State Transition Diagrams and State charts

3.5 State Transition Diagram for a Digital Watch

3.6 Let Us Sum Up

3.7 Answers for Check Your Progress

3.8 Glossary

3.9 Assignment

3.10 Activities

3.11 Case Study

3.12 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of State-Transition Diagrams
- Basic of Domain Expert Testing
- Basic of Dynamic Modelling

3.1 Introduction

A dynamic model describes about the time in which any operation is performed. It is often seen that actors are active objects where the dynamic model declares when to start or stop. Since the data stores are passive objects, so it will only react to updates and queries, hence nothing more needs to specify in case of dynamic model.

The dynamic model basically represents the time–based aspects of a system. This is related with the temporal changes in the states of the objects in a system. The main concepts are:

- State, is the situation at particular condition during lifetime of an object
- Transition, a change in the state
- Event, an occurrence that triggers transitions
- Action, an uninterrupted and atomic computation occurs due to some event
- Concurrency of transitions

A state machine models the behaviour of an object because it passes through a number of states in its lifetime due to some events as well as the actions occurring due to the events. A state machine is graphically represented by using a state transition diagram.

3.2 State-Transition Diagrams

A state generally represents

- a time period during which a predicate is true, e.g., budget – expenses >0,
- an action that is being performed, e.g., check inventory for order items
- or someone waits for an event to happen, e.g., arrival of a missing order item
- A state can be marked as "on" or "off"

When a state is "on", all its outgoing transitions are eligible to fire. For a transition to fire, its event must occur and its condition must be holding true. When a transition does fire, its action is carried out and States can have associated actions:

- OnEntry-Such actions are performed soon after state gets entered
- Do-Such actions performs during lifetime of state
- OnEvent-Such actions performs in response to event
- OnExit-Such actions occur before state exists
- Include- Such shows submachine from another state chart diagram

State Diagrams appears to be a dynamic model which describes various states by which single object will pass during its life in response to events, along with its responses and actions. There are certain features of State Diagrams:

- It describe all potential states that specific object gets into with information on how object's state changes as results of events that reaches the object.
- It is drawn for single class to show life behavior of single object.
- It acts as a condition of object at a moment in time among events.
- It shows transition which appears as relationship among 2 states showing event happening while moving from previous state to next state.
- It shows interesting events and states of object with behavior of object in reaction to event.
- It includes initial pseudo-state which automatically transitions to different state after creating an instance.
- It shows lifecycle of object, event it experiences along with transitions and states among such events.
- It is applied to many UML elements along with classes and use cases.
- It explains legal sequence of external system events which are recognized and handled by system inside the context of use case.
- The use case state chart diagram depicts system events with its sequence among use case.
- It is created for virtually any kind or class or use case.
- If an object perpetually responds the same way to an event then it's considered state-dependent with respect to that event.
- These diagrams are created for state-dependent objects having complex behavior such as Use Cases, Stateful session, systems, windows, controllers, transactions, devices and role mutators.
- External Event also known as a system event is caused by something outside our system boundary.
- Internal Event is caused by something inside our system boundary.
- Temporal event is caused by the occurrence of a specific date and time or passage of time.
- It shows External and Temporal events along with their reactions, rather than applying to design object behaviors based on internal events.
- A Transition Action is a transition that can cause an action to fire.

- A Transition Guard Condition is a transition that may also have a conditional guard or Boolean test. The transition only occurs if the test passes.
- In this, the nested states appears as sub state, where state allows nesting to contain sub states that inherits transitions of its super state.
- Actions are associated with transitions and are considered to be processes that occur quickly and are not interrupted.
- In this, the activities are linked with states which can be interrupted by some event.
- In case of transition having no event in its label, then transition occurs when any activity linked with given state is completed.
- A guard is a logical condition that will return only true or false.
- A guarded transition occurs only if the guard resolves to true.
- In this, single transition takes out of given state that guards mutually exclusive for event.
- A super state encloses all the sub states in a single state box and defines its name in a little square at the top.
- The sub states simply inherit any transitions on the super state.
- If a state responds to an event this can be put in text in the form eventName/actionName in the state box.
- When an event is generated after a period of time this is indicated with the keyword "after", for instance you may say after 20 minutes.
- An event can be generated when a condition becomes true.
- In this, special events like entry and exit, where any action marked as linked to entry event gets executed when given state gets entered by transition.
- If there is a transition that goes back to the same state with an action the exit action would be executed first, then the transition's action and finally the entry action. If the state has an associated activity as well that activity is executed after the entry action.
- Concurrent state diagrams are useful when a given object has sets of independent behaviors.
- These are good for showing behavior of object across many use cases.

- These are not very good in describing behavior involving number of objects collaborating.
- The syntax for a transition label has three parts all of which are optional: Event [Guard] / Action
- It doesn't show object interactions or collaborations

Consider examples of state diagrams:

For book class where use of entry and exit takes place:

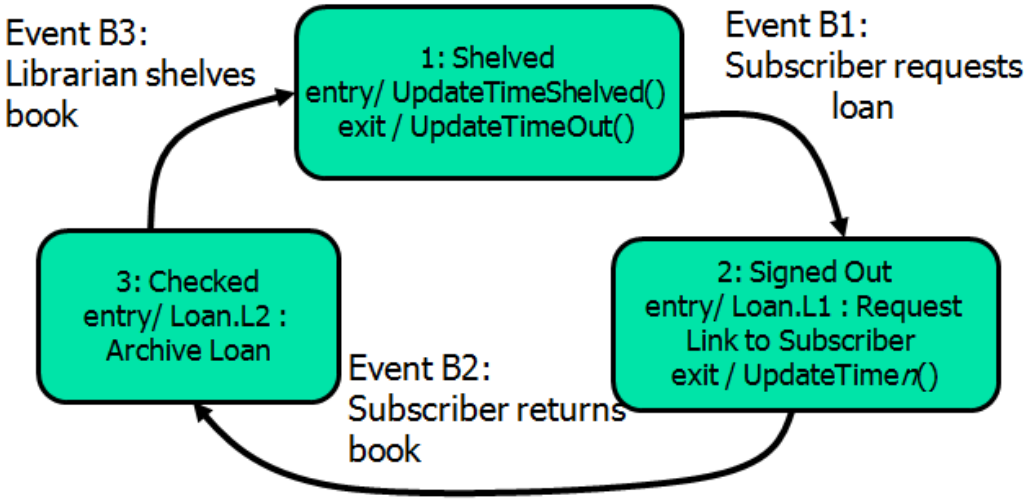


Fig 3.1 State diagram for book class

For showing life cycle of loan:

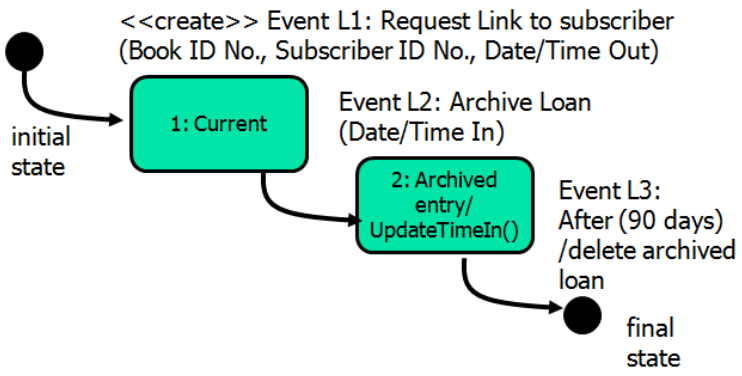


Fig 3.2 State diagram of life cycle of loan

Check your progress 1

1. What is State Diagram?

a. It describes all the potential states that a specific object will get into.

b. It is used to describe the behaviour of an object across several use cases

c. It shows the lifecycle of an object: what event it experiences and its transitions.

d. All of these

3.3 Domain Expert Testing

Domain analysis is a method for priming the pump which shows easily accessible supply of concepts which serves as classes in analysis model. After locating the domain classes, extra classes are created to show interfaces of domain model along with certain technologies like user interfaces and database which arises as domain model showing specific application.

Different development methods structure the domain model very differently. At Software Architects we use the complete UML suite of diagram types to represent the knowledge needed to describe the domain which carry set of high-level use cases showing typical system in domain application using set of guided inspection for examining domain model.

Domain models are abstract that shows as generally as possible. Constraints are stated as liberally as applicable. While this produces a model that encompasses as many applications as possible, it also makes it difficult to evaluate the "correctness" of a use of an application. A domain model may be inconsistent. By being inclusive of all potential applications, the model may contain concepts that would always appear separately in an actual application. The use of the inheritance notation in UML supports the inclusion of alternatives. A car contains a single transmission but the domain model should present all possible types of transmissions that are used.

The testing process has two roles:

- modelling expert
- domain expert

Domain expert caters information related to domain where it shows expertise in syntax and semantics. The modelling expert is familiar with guided inspection process which serves as driving force for testing process. Figure 3.3 shows that single modeling expert manages many domain experts that works with model development and model testing processes.

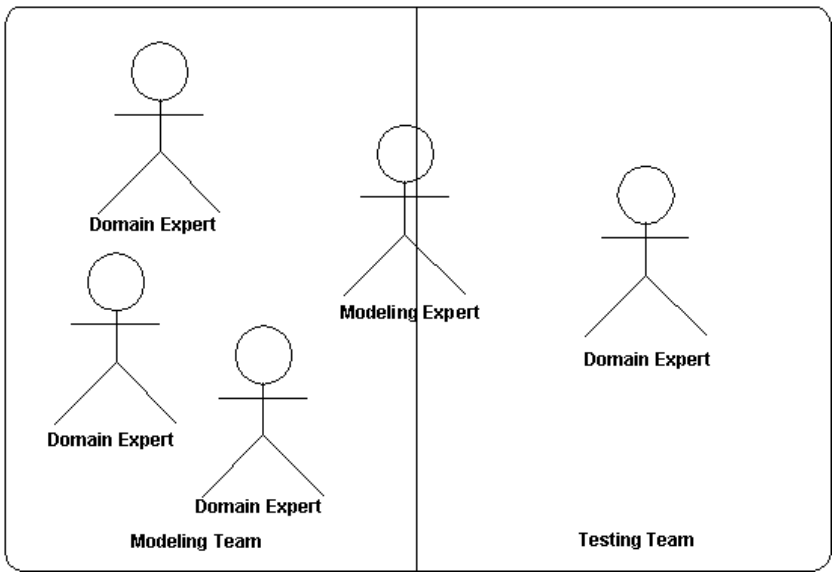


Fig 3.3 single modelling expert

The model testing process is tightly attached with model development process which iterates within modelling process by periodically switching from modelling activity to testing activity. This provides quick feedback and often provides new information to be modelled. The basic steps are the same as for any testing process:

Analyze - Much of the testing analysis has been done if the use case descriptions contain sufficient information to allow them to be prioritized.

Construct - Write scenarios from the use cases. Vary the number of test cases based on the use case priorities.

Execute and Evaluate – The test session involves role-playing in which the modellers and developers step through the model.

Check your progress 2

1. Model testing involves:

a. modelling activity

b. testing activity

c. both a and b

d. neither a nor b

3.4 Dynamic Modelling - State Transition Diagrams and State charts

Actions can be related to getting into an exciting a state as an alternative to connecting them to transactions. An entry action is performed when any transition enters the state and an exit action is performed once a state is exited. This enables a state to be expressed in terms of matched entry and exit actions without regard to what happens before a state becomes active.

An internal action doesn't change the state it executes at intervals the state. Automatic transactions fire and change the state once their conditions are met and any activity in the current state is terminated.

Sending Events:

An object can send an event {to another to a different} object together with an attribute. A race condition occurs once a state might accept events from more than one object. In this case the order of the events becomes important since it'd have an effect on the final state of the object.

Synchronization of concurrent activities:

Sometime the object might perform 2 or additional activities at a time at the same time. The internal steps might not be synchronous, but both the activities should be completed before the object will progress to next state. Any transition into a state with sub diagrams activates each of the diagrams.

State Diagram Elements

In order to check an interrupt obtained from device drivers, the data is assigned to the hardware which will work correctly if the data is less. The elements of state diagrams are:

- Pseudo state – is the starting point of a state chart
- State – is the condition that occurs during an object's life when it satisfies some criteria, performs some action, or waits for an event
- Transition – is the movement of object from one state to another state
- Message event – is the trigger for the transition

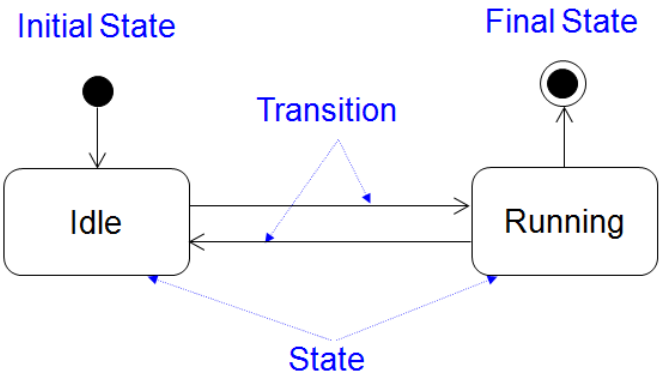


Fig 3.4 State Diagram

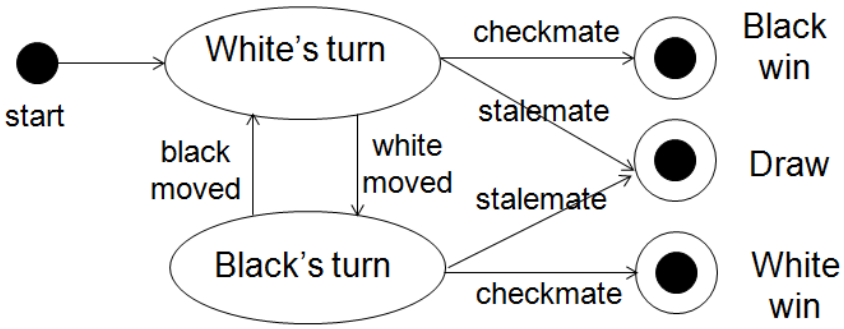


Fig 3.5 Initial and final State

Check your progress 3

1. When the exit action does is performed?
   a. When a transaction enters the state
   b. When the transaction leaves the state
   c. When the task is performed
   d. None of these

3.5 State Transition Diagram for a Digital Watch

In a simple digital watch, showing display and two buttons for setting, such as button A and B. The watch carries two modes of operation:

- Display time mode: Displays hours and minutes separated by flashing colon.
- Set time mode: Displays set hours and minutes.

In this, button A selects modes where each time on pressing, advances mode in sequence: display, set hours, set minutes, display etc, while button B advances hours or minutes once each time when pressed as shown in figure 3.6.

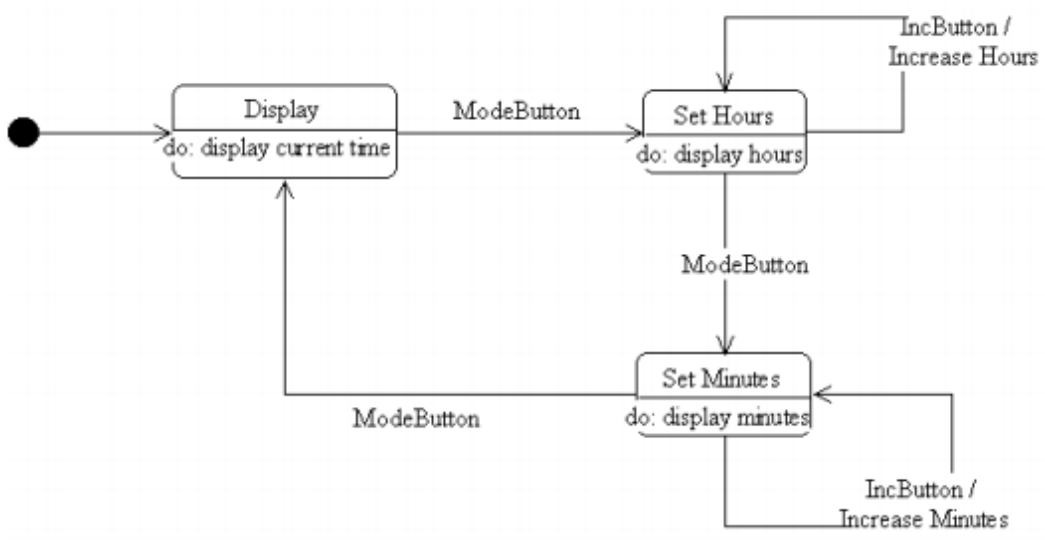


Fig 3.6 digital watch

In this, there are three states:

- Display
- Set Hours
- Set Minutes

Display is the start state, shows by arrow from black dot. Here, Set Hours state, event Mode Button causes a transition to Set Minutes state, whereas event Inc Button causes the action Increase Hours to occur.

Check your progress 4

1. in digital clock, which among the following is not a state:

   a. Display

   b. Set Hours

   c. Set Minutes

   d. Set Days

3.6 Let Us Sum Up

While studying this unit, we have learnt that events are occurrences which trigger state transition of object or group that carries location in time and space without period of time related to it.

It is noted that state diagrams are dynamic model showing various states by which single object which passes during its life in response to events, along with its responses and actions.

It is found that to check an interrupt from device drivers, data gets assigned to hardware which will work correctly if the data is less.

It is found that concurrency appears to be tendency for things which happens at same time in a system where many things happens at same time.

It is noted that dynamic model shows time in which an operation is performed where actors are active objects where dynamic model declares when to start or stop.

3.7 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d)

Check your progress 2

Answers: (1 - c)

Check your progress 3

Answers: (1 – b)

Check your progress 4

Answers: (1 – d)

3.8 Glossary

1. **Design -** In modelling, problem related to object domain in programming.
2. **Objects design -** It is design strategy where system designers think of operations or functions.
3. **Algorithm -** It is a stepwise procedure which solves problem in required manner.
4. **State -** In modelling, state refers to time period during which predicate is true.

3.9 Assignment

What is the purpose of state diagrams?

3.10 Activities

What are concurrent and nested state diagrams?

3.11 Case Study

Develop steps to test transition diagram.

3.12 Further Reading

1. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
2. John D. McGregor. The Fifty Foot Look at Analysis and Design Models, Journal of Object-Oriented Programming, July/August 1998.

Block Summary

In this block, you have learnt and understand about the basic of UML Use Case Diagram along with necessary steps. The block gives an idea on the study and concept of Use Case Diagram. You have been well explained on the concepts of state diagrams and basic concept on dynamic modeling.

The block detailed about the basic of various class definition along with responsibilities to object methodology. The concept related to events and its features in dynamic modelling are also explained to you. You will be demonstrated practically about drawing of effective UML Diagrams.

Block Assignment

Short Answer Questions

1. What is Collaboration Diagrams?
2. Explain UML Package Diagram?
3. Write note on Domain Expert Testing?
4. Write short note on State Transition Diagrams?

Long Answer Questions

1. Write short notes on UML Collaboration Diagram?
2. Write short note on UML Component Diagram?
3. Write note on considering effective UML Diagrams?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any Other Comments

……

……

……

……

……

……

……

……

……..

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
'Jyotirmay Parisar', Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi,
Ahmedabad-382 481.

OBJECT ORIENTED ANALYSIS AND DESIGN

BCA - 502

BLOCK 4:
COMPONENT TECHNOLOGY AND CASE STUDIES

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED ANALYSIS AND DESIGN

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Prof. K J Sharma

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Mr. Prashant Tikone
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED ANALYSIS AND DESIGN

Contents

BLOCK 1: INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND METHODOLOGIES

UNIT 1 INTRODUCTION TO OBJECT ORIENTED ANALYSIS AND DESIGN

Introduction, Mechanism Of Object-Oriented Approach, Tools And Approaches In OO Analysis And Design

UNIT 2 THE OBJECT PARADIGM

Introduction, Getting Acquainted With The Class Project, Object-Oriented Conceptualization, The Software Life Cycle

UNIT 3 OBJECT ORIENTED METHODOLOGIES

Introduction, Object Oriented Methodologies, Object Process Methodology (OPM)

BLOCK 2: OBJECT MODELLING

UNIT 1 REVIEW OF OBJECT MODELING

Introduction, Principles of Modeling, Object-Oriented Modeling, The Use Case Model

UNIT 2 IMPORTANCE OF MODELING

Introduction, Importance of Modeling, Importance of Business Domain Modelling

BLOCK 3: UML AND OTHER DIAGRAMS

UNIT 1 UNIFIED MODELLING LANGUAGE (UML)

Introduction, Unified Modeling Language (UML) Basics, Practical UML: A Hands-On Introduction For Developers, UML Package Diagram, UML Object Diagram, UML Use Case Diagram, UML Sequence Diagram, UML Collaboration Diagram, UML State chart Diagram, UML Activity Diagram, UML Component Diagram, UML Deployment Diagram, Tips For Effective UML Diagrams

UNIT 2 INTERACTION DIAGRAMS

Introduction, UML Collaboration Diagram, Collaboration Diagrams

UNIT 3 STATE TRANSITION DIAGRAMS

Introduction, State-Transition Diagrams, Domain Expert Testing, Dynamic Modelling - State Transition Diagrams And State charts, State Transition Diagram For A Digital Watch

---

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

UNIT 1 INTRODUCTION TO COMPONENT TECHNOLOGY

Introduction, Component, Component Characteristics, Components And Objects, Component Extraction From Legacy Software, Software Components – Benefits, Need For Component-Based Development Approach, Different Standards For Component Software, JavaBeans

UNIT 2 CASE STUDY

UML Diagrams Library Management System, Problem Statement, Online Mobile Recharge

---

Dr. Babasaheb
Ambedkar
Open University

BCA - 502

OBJECT ORIENTED ANALYSIS AND DESIGN

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

UNIT 1
INTRODUCTION TO COMPONENT TECHNOLOGY 02

UNIT 2
CASE STUDY 15

BLOCK 4: COMPONENT TECHNOLOGY AND CASE STUDIES

Block Introduction

Component technology is basis of software design and development that is mainly linked with Object technology where components are typically business objects which are predefined and reusable in behaviours. Implementation detail is hidden in the interfaces that isolate and encapsulate set of functionality.

In this block, we will detail about the basic of Software components and related objects. The block will focus on the study and concept of Component-Based Development Approach. You will get an idea on DMA control input output and basic programmed input output.

In this block, you will make to learn and understand about the basic of UML Diagrams with its designing techniques. The concept related to JavaBeans and its features will also be explained to the students. You will be demonstrated practically about Online Mobile Recharge state diagrams.

Block Objective

After learning this block, you will be able to understand:

- About Component Characteristics
- Basic of Component Technology
- Features of Software Components
- Concept of Need For Component
- Detailed about JavaBeans

Block Structure

Unit 1: Introduction to Component Technology

Unit 2: Case Study

UNIT 1: INTRODUCTION TO COMPONENT TECHNOLOGY

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Component

1.2.1 Component Characteristics

1.2.2 Components and Objects

1.3 Component Extraction from Legacy Software

1.4 Software Components – Benefits

1.5 Need for Component-Based Development Approach

1.6 Different Standards for Component Software

1.7 JavaBeans

1.8 Let Us Sum Up

1.9 Answers for Check Your Progress

1.10 Glossary

1.11 Assignment

1.12 Activities

1.13 Case Study

1.14 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of Components and Objects
- Benefits of Software Components
- Basic of JavaBeans

1.1 Introduction

The days of large, monolithic software systems are fast moving into oblivion. The pace of software development becomes aggressive with development cycles reduced drastically. The current trend favours a short-term development process where large and complex applications are being built using a series of smaller parts, referred to as components. Component technology is the next step in the evolution of software design and development. It is strongly associated with Object technology, though this association is not necessarily an accurate one.

Component-based software development is based on building software systems from previously-existing software components. In CBD, reuse of common parts in component form can reduce the development cost of new systems, and reduce the maintenance cost associated with the support of these systems. However, existing programs have usually been built using another paradigm, such as the object-oriented (OO) paradigm. OO programs cannot be reused rapidly or effectively in the CBD paradigm even if they contain reusable functions.

1.2 Component

Future systems shall be developed by assembling co-operative software units. These units need not necessarily originate from the same vendor, but will conform to a standard interface for units offering their respective functionality. Assembly of such units will be aided by use of tools which will extract self descriptive information from these units. Delivery of such an assembled system will involve the deployment of these units configured appropriately. These units may be delivered on any platform. Such software units are known as components.

Components are typically business objects that have predefined and reusable behaviours. Implementation detail is hidden in the interfaces, which isolate and encapsulate a set of functionality. Interfaces are the means by which components connect. An interface is a set of named operations that can be invoked by clients. More importantly, well-defined interfaces define the component's entry points, and a component's accessibility is done only via its interface. In a component-based approach, providers and clients communicate via the specification of the interface, which becomes the mediating middle that lets the two parties collaborate and work together.

1.2.1 Component Characteristics

The component characteristics are:

- It is independent application level software
- It is designed for particular purpose and not for particular application
- It is a self-contained and pluggable abstract data types having grained entities
- It can be accessed by way of well defined interfaces
- Its interface serve as access points to its functionality that are done by objects internally
- In a system there appears only single instance of component.
- It uses object reference to handle object for implementing interface.
- It carries systems having typical feature components from various vendors
- It is interoperable by conformance to industry standard interfaces
- It is encapsulated and modular
- It uses inheritance for implementing reasons which is remain intact to component clients
- It cannot be extended by inheritance.

1.2.2 Components and Objects

It is worth considering the relationship existing between components and objects. A component approach to software development builds upon fundamental constructs of object-oriented paradigms. Although the terms "component" and "object" are often used interchangeably, a component is not an object. An object is an instance of a class that gets created at runtime. A component can be a class, but is usually a collection of classes and interfaces. At runtime, a component becomes alive when its classes get instantiated. Therefore, at runtime, a component becomes a web of objects.

Their usage seems synonymous. While objects are well-suited for component construction, a non-OO approach to component construction is perfectly valid. The following explains how objects differ from components.

- Components are large-grained entities, e.g. a payroll module. Objects are fine-grained entities, e.g. a Person.
- Components are accessed via well-defined interfaces. Objects are also accessed via interfaces. Thus the two are similar in this respect, but not identical. Components interfaces are based on industry standards, object interfaces are not. A component with a CORBA IDL interface is easily accessible from any language. An object implemented in C++ is not easily accessible from another language. Components need not use OO approach to implement their interfaces. The same is also true of objects. The objects often act as wrappers to non-OO-code.
- Only single component instances will exist per system whereas multiple objects will exist per system. Components are more akin to Object factories which also delegate functionality to objects within the factory.
- Component based systems will be made up of heterogeneous vendor components. Seamless integration is possible as components will be built to industry standard interfaces. Achieving the same with objects tends to be more difficult as objects are built to proprietary interfaces, thus can not be seamlessly integrate. Such integration is possible if the object interfaces are standard such as Java AWT.
- Components will be runnable out of the box. Only Java objects on a system running a JVM can boast of this property.
- Components can be encapsulated and non-extendable. Objects will be both encapsulated and extendible through inheritance. But inheritance may be misused and hence this is being avoided in components as component assemblers are experts in domain knowledge not on OO techniques.

Check your progress 1

1. In component diagrams, building block showing two rectangles on left is classified as:

   a. component types

   b. interfaces

   c. dependency relationships

   d. all of above

2. Attribute of object carry information about:

a. state

b. method

c. behaviour

d. procedures

1.3 Component Extraction from Legacy Software

There are a whole lot of exciting and robust components in the existing systems. Extraction of them and reusing for the purpose of building software applications is one interesting task. The other one is to build components from the scratch using component-based programming languages.

Component-oriented programming addresses the aspects of programming components. Component construction itself can be performed using arbitrary programming languages, as long as the language supports the particular component standard's interface conventions. Many programming languages such as COBOL, C, C++, Pascal, Smalltalk lack support for encapsulation, polymorphism, type safety or a combination of these. So nowadays Java is becoming the most favoured programming language for building software components.

In the absence of existing legacy code, a component must be built from scratch. In this case, it makes sense to implement a component using an OO approach. However, it is often the case that legacy code providing similar functionality will be available somewhere.

The aim of components is to deliver better applications to the market in a shorter time. This is easily achieved through leveraging existing components. Thus the philosophy of component assembly is re-use. The philosophy for component construction remains the same.

When assembling a component based system, it is assumed that the selected components are fit for the purpose. That is, the components have been rigorously tested and their performance metrics well understood. Reuse is encouraged through faith in the components. When constructing components, existing code will be often be reused. We below give a brief explanation for some of the approaches in this regard.

Check your progress 2

1. What are the benefits for component based product lines?

   a. Being able to take faster advantage of new product and new technology

   b. Higher employee productivity

   c. Increase in time to market

   d. All of the mentioned

1.4 Software Components – Benefits

The benefits of software components are:

- A component is more generalized and application-independent
- Components can be reused in a variety of contexts.
- Individual components can be custom-made to suit specific requirements, whereas existing ones can be customized to meet those requirements.
- Several components can be assembled to form a functioning system.
- Upgrades of individual components end the problem of massive upgrades as done in monolithic systems
- Components can live anywhere on a network - in computers best suited for running them - based on the functions the components provide.
- Using distributed computing system standards such as CORBA, Java RMI and EJB, and Microsoft's COM/DCOM, components can be distributed among the computers in an enterprise network.

Check your progress 3

1. Which among the following does not represents distributed computing system standards?

   a. UNIX

   b. CORBA

   c. COM

   d. JAVA

1.5 Need for Component-Based Development Approach

The most critical issue in the delivery of the ICT solutions in 21st century will not be hardware but software, applications which support new forms of business like Web Enterprises. The developments in hardware and networking will provide enough processing, storage and communication capability. However with the current approach of software development it will take many years to deliver the applications that really support reinvention of the business and take maximum advantage of the capabilities of the information and communication technology.

Software development is a craft industry, where programmers build tailor-made applications from beginning. The problems are the long development time and high costs for new applications and the reality that the resulting systems usually do not fully meet the business requirements. These difficulties affect even more the adaptation of existing systems. Development time and costs of modifications tend to grow over the years. Eventually it becomes impossible to meet the changing business requirements with the existing system. A total replacement with a new system is necessary. Solutions for these problems are sought in packages or in better methods and tools.

Application packages are an alternative, but still need costly implementation projects and adaptation of the business processes to the functionality of the packages. Packages have their own architecture, which is not always adjustable to the structure of the business. Normally a package does not support all business functions in the supply chain. So it is necessary to extend the functionality with other packages or tailor-made applications. Packages do not integrate well with existing applications and the package may need ICT infrastructure other than the current installed one. Maintenance of existing package-based systems is still expensive and time-consuming especially with regard to the necessary upgrading caused by new releases.

Methods and tools have streamlined the design and development process and realised a first industrialisation of the application delivery. But methods and tools did not improve the adaptability to the business of the resulting applications. Reason is that methods and tools strongly focus on the improvement of the development process and not focus on an adaptable construction of the applications.

The only way to bridge the application gap is to reinvent the construction of applications. The car industry is a good example. This industry realised a shorter

life cycle of design, development and production of their products through standardisation of the product components. The car industry is also able to mass-produce cars that within certain limits can be customised to the wishes of the client.

Application developers also need an application construction, which is based on components. Cap Gemini believes that the solution for coping with the challenge of application development in the 21st century lies in a new approach of the application construction: Component-Based Development.

Component-Based Development is a new and rapidly emerging approach for application development by plugging together previously built components. This paradigm is not something new. It has already been endorsed by many industries.

Most application developers do have some notion of the essence of a component. It has encapsulated functionality and a well-defined interface providing some service. But in practice a wide variety of different types of components exist, because the value of components depends on the beholder. For example for the IT engineer components is a persistency manager for objects or a transaction manager for secure transactions. For a business architect components are a pricing pattern or a financial product. An application developer is more interested in components like an order window. Apparently the concept of componentisation applies to all levels of abstraction, from business models to technical components and from design models to software coding.

However, the way that all these different types of components can be adapted to meet specific requirements and how they interact with other components has not yet been demystified. The solution in our opinion is by no means just a matter of precise specification of a single component. The solution also is not an approach where developers freely choose their components and simply have a working software system in the end. Components should not be regarded as solitaire phenomena. Instead, they must be bound to a specific context to ensure proper construction and operation.

Components are designed to interact with its environment through its well-defined interfaces but to encapsulate their implementation. Component-based software development brings the potential for:

- Significant reduction in the development cost and time-to-market of enterprise software systems because developers can assemble such systems from a set of reusable components rather than building them from scratch,

- Increasing the reliability of enterprise software systems - each reusable component undergoes several review and inspection stages in the course of its original development and previous uses, and cbsd relies on explicitly defined architectures and interfaces,
- Improving the maintainability of enterprise software systems by allowing new, higher-quality components to replace old ones, and
- Enhancing the quality of enterprise software systems - application-domain experts develop components, then software engineers who specialize in component software engineering assemble those components into enterprise software systems.

Check your progress 4

1. Component based software development is good for:

   a. lowering of development cost

   b. using components again

   c. increases reliability of enterprise software systems

   d. all of above

1.6 Different Standards for Component Software

Standards are format which are approved by recognized standards organization or established as de facto standard by industry for the purpose of:

- programming languages
- operating systems
- data formats
- communications protocols
- electrical interfaces

Standards are essential in the production of electronic resources because they facilitate data interchange and representation and management of information. That they should be used is almost unarguable; which standards should be used is another question.

From a user's standpoint, standards are extremely important in the computer industry because they allow the combination of products from different manufacturers to create a customized system. Without standards, only hardware and software from the same company could be used together. In addition, standard user interfaces can make it much easier to learn how to use new applications.

In the networked environment, the ability to easily share information is crucial. Central to information sharing is the software environment, particularly software used for word processing, spreadsheets, databases, network browsing, and electronic mail. Developing software standards, will greatly improve functions between these systems. Standards can facilitate exchange of information.

Software standards enable software to interoperate. There is not consensus on what the standards should include. Software standards are one of the unsolved problems in software engineering. There are multiple reasons behind software standards such as safety, economic and social reasons. on-standard implementation of standards or specifications by multiple organizations results in a requirement for implementation specific code and special case exceptions as a necessity for cross-platform interoperability. Notable modern examples include web browser compatibility and web-services interoperability. The arbitrariness of most software concepts, which is related to historical hardware and software implementation, lack of common standards worldwide, and economic pressures.

Open Standard

An open standard is a standard that is publicly available and has various rights to use associated with it. The terms "open" and "standard" have a wide range of meanings associated with their usage. The term "open" is usually restricted to royalty free technologies while the term "standard" is sometimes restricted to technologies approved by formalized committees that are open to participation by all interested parties and operate on a consensus basis.

The purpose of an open standard is to increase the market for a technology by enabling potential consumers or suppliers of that technology to invest in it without having to either pay monopoly rent or fear litigation on trade secret, copyright, patent, or trademark causes of action. No standard can properly be described as "open" except to the extent it achieves these goals.

Check your progress 5

1. The benefit of Standards in Software is:

   a. more publicity

   b. durability

   c. cording with same hardware's

   d. coordination with different hardware's

1.7 JavaBeans

JavaBeans is the software component architecture for the Java language. A Java Bean is a reusable software component which manipulates visually in builder tool. It is an ordinary Java classes that follow certain conventions such as:

- property of Java Bean class is implemented by defining "get" and "set" functions
- operations of Java Bean class are other public functions defined in class

Creating Java Bean

- Initially design ordinary Java class with related properties and public operations which supports
- Ensure that every property carry "get" and "set" operation in public interface of Java Bean class
- Ensure that class conforms to Java 1.1 event model
- Class being visual Bean should be derived class of AWT classes
- Use some other helper classes
- Bundle .class files for main Java Bean class with helper classes in Java Archive file

Java Beans make good interfaces

- It provides interface to other non-Java applications which uses Microsoft's ActiveX components
- It shows a bridge among Java Beans and ActiveX
- It encapsulates interface to external database

- In this, get and set operations uses Java database interface classes to interact with relational database
- It is used to bring together the applications with application builder tool

Check your progress 6

1. JavaBeans uses:

   a. Linux

   b. Java 1.1

   c. COM

   d. all of these

1.8 Let Us Sum Up

In this unit we have learnt that component based software development depends on building software systems from earlier existing software components where common parts in component are used again to form development cost of new systems by lowering maintenance cost along with support of systems.

It is studied that components are business objects which were earlier defined and reusable behaviours.

Component-oriented programming describes aspects of programming components which are framed itself using arbitrary programming languages.

JavaBeans is software component architecture for Java language that can be used again which manipulates visually builder tool.

1.9 Answers for Check Your Progress

Check your progress 1

Answers: (1 - d), (2 – a)

Check your progress 2

Answers: (1 - c)

Check your progress 3

Answers: (1 – a)

Check your progress 4

Answers: (1 - d)

Check your progress 5

Answers: (1 - d)

Check your progress 6

Answers: (1 – b)

1.10 Glossary

1. **Component-based software -** Previously developed software systems which can be reused to lower development cost of new systems.
2. **Components -** Business objects which isolate and encapsulate functionality.
3. **JavaBeans -** Java language software component that can use again and again for manipulating visually in builder tool.

1.11 Assignment

Explain the Component Technology?

1.12 Activities

Study about requirement of Component-Based Development Approach.

1.13 Case Study

Discuss about Component Extraction From Legacy Software.

1.14 Further Readings

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

UNIT 2: CASE STUDY

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 UML Diagrams Library Management System

2.3 Online Mobile Recharge

2.4 Let Us Sum Up

2.5 Answers for Check Your Progress

2.6 Glossary

2.7 Assignment

2.8 Activities

2.9 Case Study

2.10 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Concept of UML Diagrams
- Understand about Online Mobile Recharge

2.1 Introduction

UML is a graphical modelling language that supposedly unifies and integrates the various notations used before by the various methodologies proposed. It is the requirement as amount of object-oriented methods increased from less to much higher. It constitutes the required standard notation and semantics for correctly describing software built with object oriented or component-based technology. It's undoubtedly a step in the right direction; however it's not an ideal or universal modelling language. we tend to believe that UML, as it stands today, must be, in some contexts and for a few application domains, complemented with alternative meta-models or a minimum of adapted to address those meta-models.

2.2 UML Diagrams Library Management System

Library Management System

The case study of Library Management System is library management software which is applied for monitoring and controlling transactions which exists in library. The case study gives detailed description on library and daily transactions. In order to maintain record of new and retrieve details about books present in library, the details regarding basic operations such as:

- adding new member
- new books
- searching books
- action of members in terms of borrow and return of books

The case study features familiar and well thought-out attractive user interface which combined searching, insertion and reporting details. The study focus on report generation facility of library system that helps in having good idea about members. The description shows functions achieved using case study:

End-Users:

- Librarian: To maintain and update the records and also to cater the needs of the users.
- Reader: Need books to read and also places various requests to the librarian
- Vendor: To provide and meet the requirement of the prescribed books.

Class Diagram

Classes identified:

- Library
- Librarian
- Books Database
- User
- Vendor

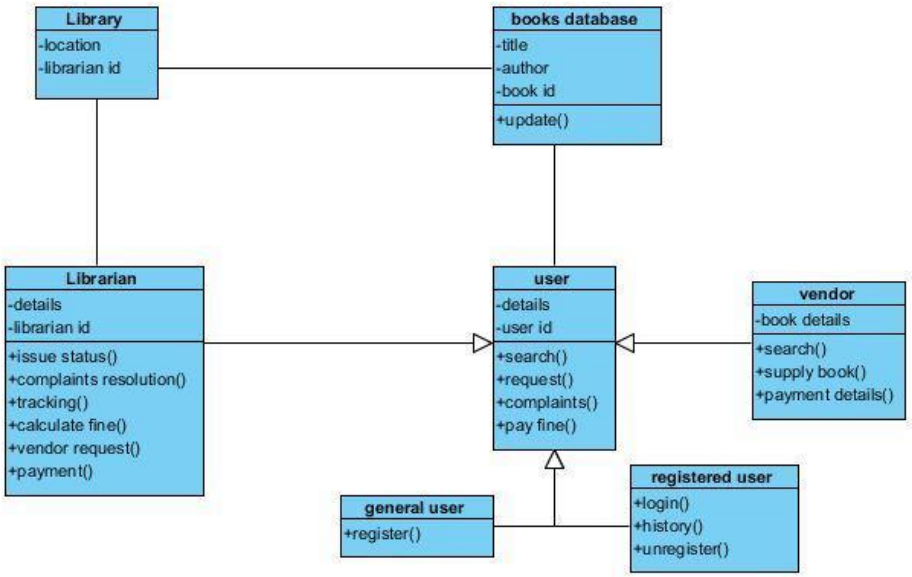


Fig 2.1 Class Identification

Use-case Diagram

Actors vs Use Cases:

Librarian

- Issue a book
- Update and maintain records
- Request the vendor for a book
- Track complaints

User

- Register
- Login
- Search a book
- Request for issue
- View history
- Request to the Librarian
- Unregistered

Books Database

- Update records

- Book status

Vendors

- Provide books to the library
- Payment acknowledgement

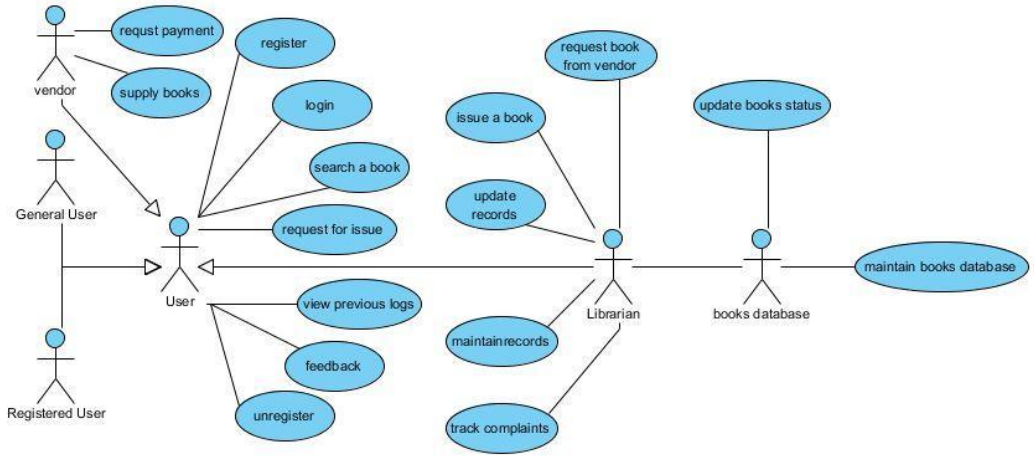


Fig 2.2 Use Case Diagram

Sequence Diagram

Figure 2.3 shows sequence diagram which describes searching books process of issuing books as demanded by user from librarian:

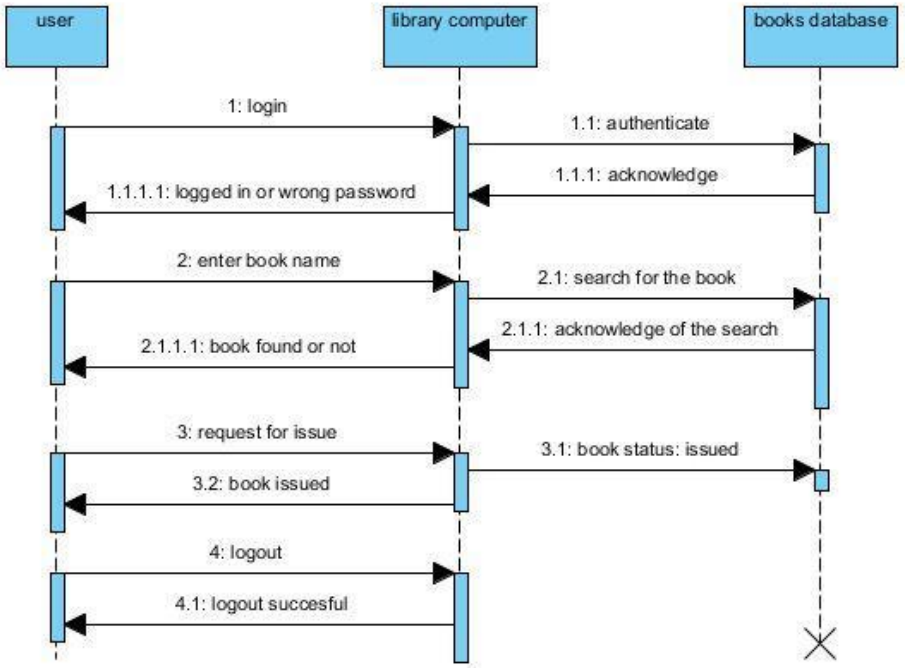


Fig 2.3 Sequence Diagram

Collaboration Diagram

Figure 2.4 shows description of collaboration diagram which is applied in order to locate book and show process of issuing it as demanded by user from librarian:

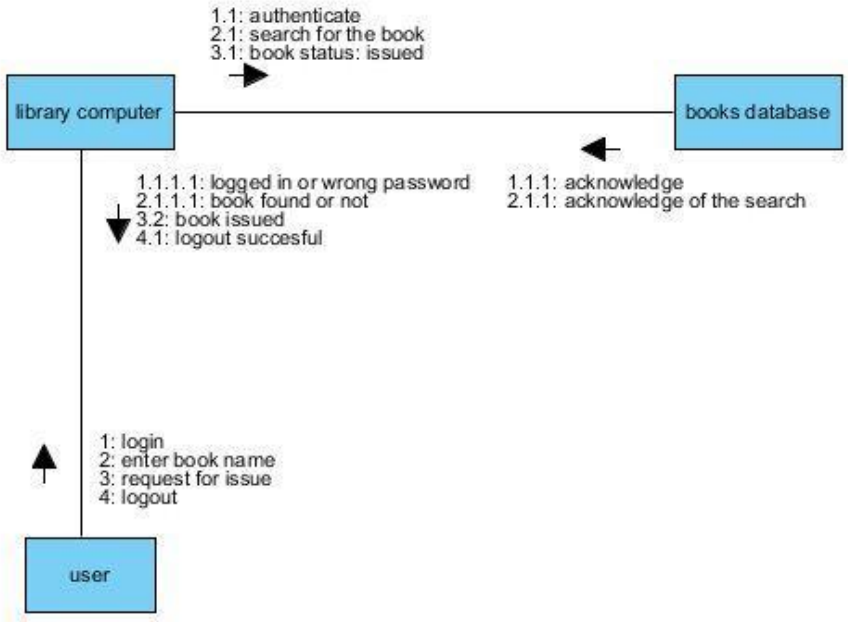


Fig 2.4 Collaboration Diagram

Activity Diagram

Activities:

- User Login and Authentication
- Search book operation for Reader
- Acknowledge and Issue books to the users by the Librarian
- Provide books requested by the Librarian from the Vendor
- Bill payment from the Librarian to the Vendor
- Status of the books updated in the Books Database

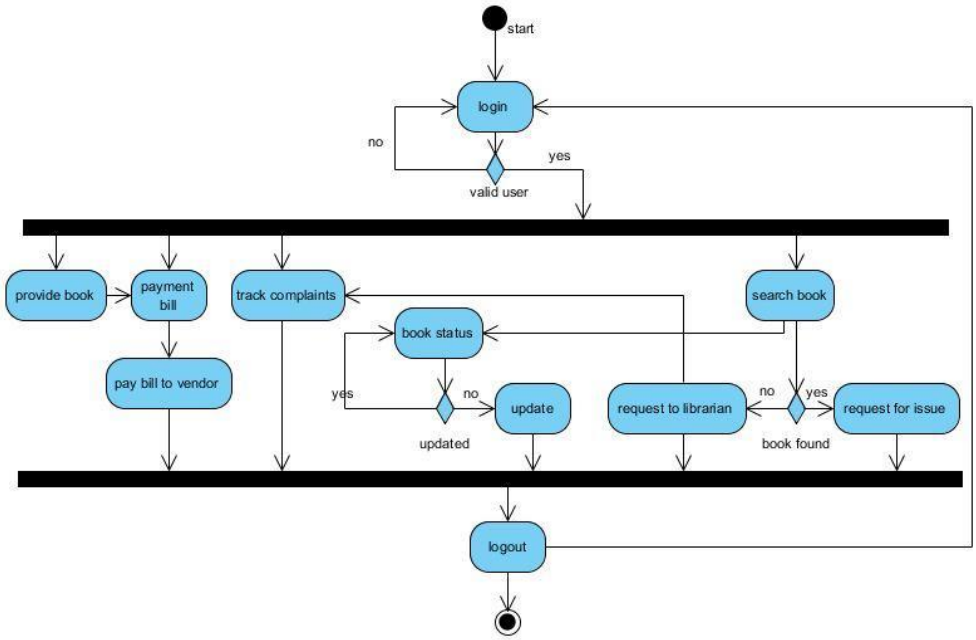


Fig 2.5 Activity Diagram

State Chart Diagram

States:

- Authentication
- Successfully logged on or re-login
- Book search by user / request vendor (librarian) / provide request book (vendor)
- Receive acknowledgement
- Logged off / re-search / new function

Transitions:

- Authenticate = Logging in
- Logged in ---> Search <---> Acknowledgement
- Logged in ---> Request Vendor <---> Provide Book <---> Acknowledgement
- Logged in ---> Provide Book <---> Acknowledgement
- Acknowledgement ---> Logged off

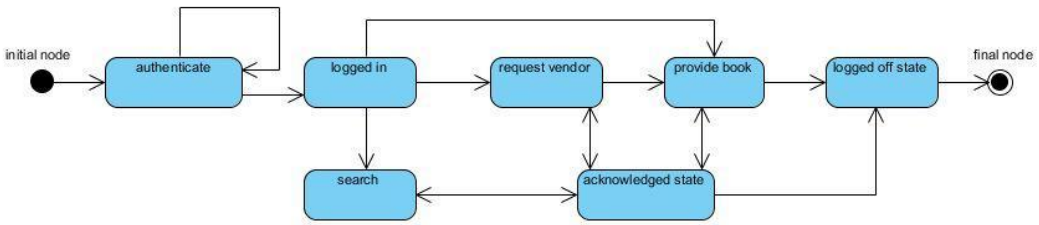


Fig 2.6 State Chart Diagram

Component Diagram

Components:

- Register Page (visitor / vendor)
- Login Page (user / librarian / vendor)
- Search Page (user / librarian / vendor)
- Request Vendor Page (librarian)
- Request Book Issue Page (user / vendor)
- Issue Status Page (librarian)
- Make Payment Page (librarian / vendor)
- Provide Books Page (librarian)
- Logout Page (user / librarian / vendor)

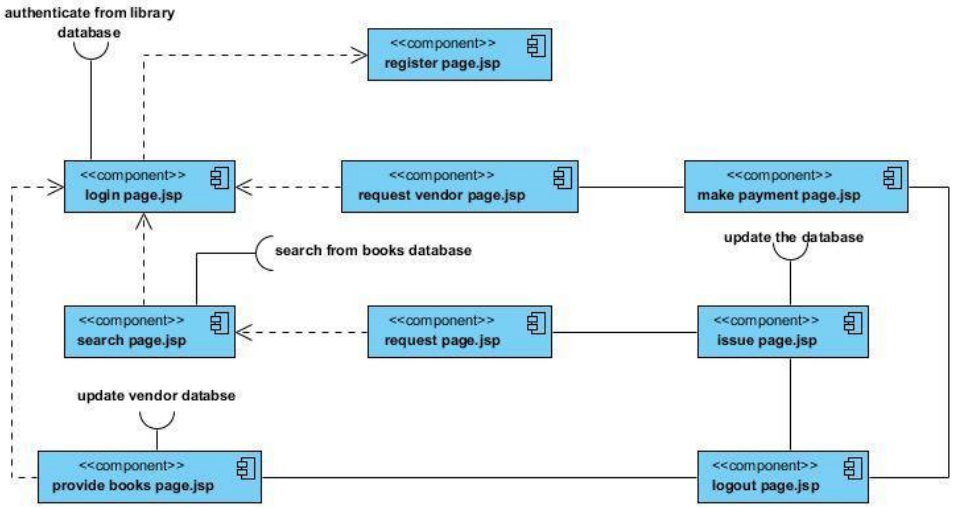


Fig 2.7 Component Diagram

Deployment Diagram

Systems Used:

Local Consoles / Computers for login and search purposes by users, librarian and vendors.

Library LAN Server interconnecting all the systems to the Database.

Internet to provide access to Vendors to supply the requested books by the Librarian

Vendor Server to maintain the records of the requests made by the librarian and books provided to the library.

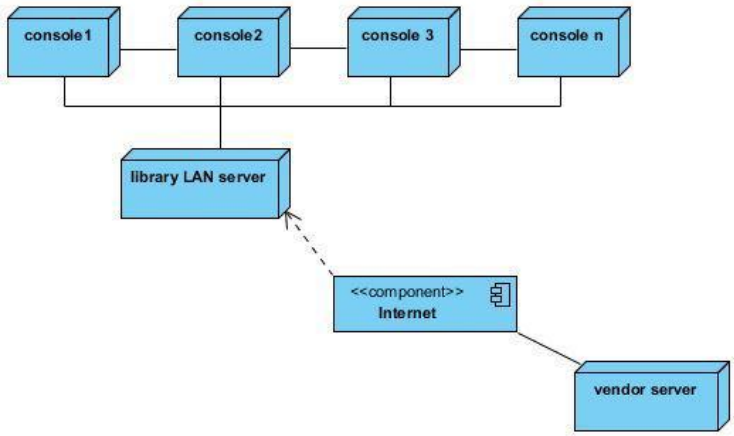


Fig 2.8 Deployment Diagram

Check your progress 1

1. What are the advantages of using object modelling?

   a. It models the problem domain objects

   b. It can easily mapped the records to corresponding database table

   c. It is used as return values for data Access Object methods

   d. All of these

2.3 Online Mobile Recharge

This case study involves in Online Mobile Recharge which shows detailed regarding activities of mobile service providers. It shows how such application detailed about information on mobile service provider which can be in areas of:

- Plans
- Options
- Benefits

The study provides information related to several schemes and services which are given by company to its customers where he/she can select any plan, recharge option, bill date and details regarding any mobile recharge application. The advantage of such study facilitate customer to have recharging facility of any service provider in same platform.

- End users:
- Service Provider:

Service Provider is any company who provide mobile services to customers related to mobile connections along with applications related to mobile recharging and several plans. Request comes from customers which further gets verified at admin side that helps in checking balance and customers request for certain mobile services and recharge application.

Third party System Administrator:

Administrator is somebody who monitors every mobile user and its transactions and handles the Service Providers in terms of managing user accounts. On putting by user, the admin will check for available balance in user account, and afterward request to Service Provider for processing of information. Admin carry every information about user that relates to schemes and recharge options.

User:

There are 2 categories in the user Module:

- Registered User
- Visitor

Any customer who needs to use Online Mobile Recharge services at given time from any where should get registered and afterward can recharge its mobile online or from any counter. Visitor is one who visits Online Mobile Recharge application and after filling required information about Service Providers will able

to recharge mobile online after entering all bank details or debit/credit card details.

Class Diagram

Classes Identified:

- User: Registered, Visitor
- Third Party System Administrator
- Third Party Server/ Database
- Service Provider
- Direct or Non- Third Party User

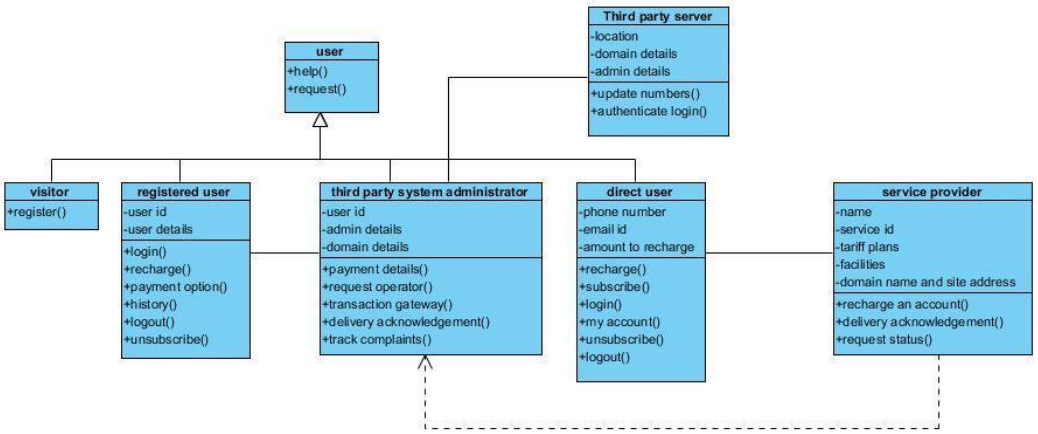


Fig 2.9 Class Diagram

Use-Case Diagram

Actors vs Use Cases:

User

- Register.
- Recharge.
- Select Payment Gateway.
- Select service Provider.
- Make payment.
- Third Party Administrator
- Forward User request to Service Provider.
- Track Complaints.

Third Party Server/ Database

- Authenticate the Registered users.
- Maintain the Log.

Service Provider

- Recharge the user requested either directly or through the third party system.
- Provide various plans to the user.

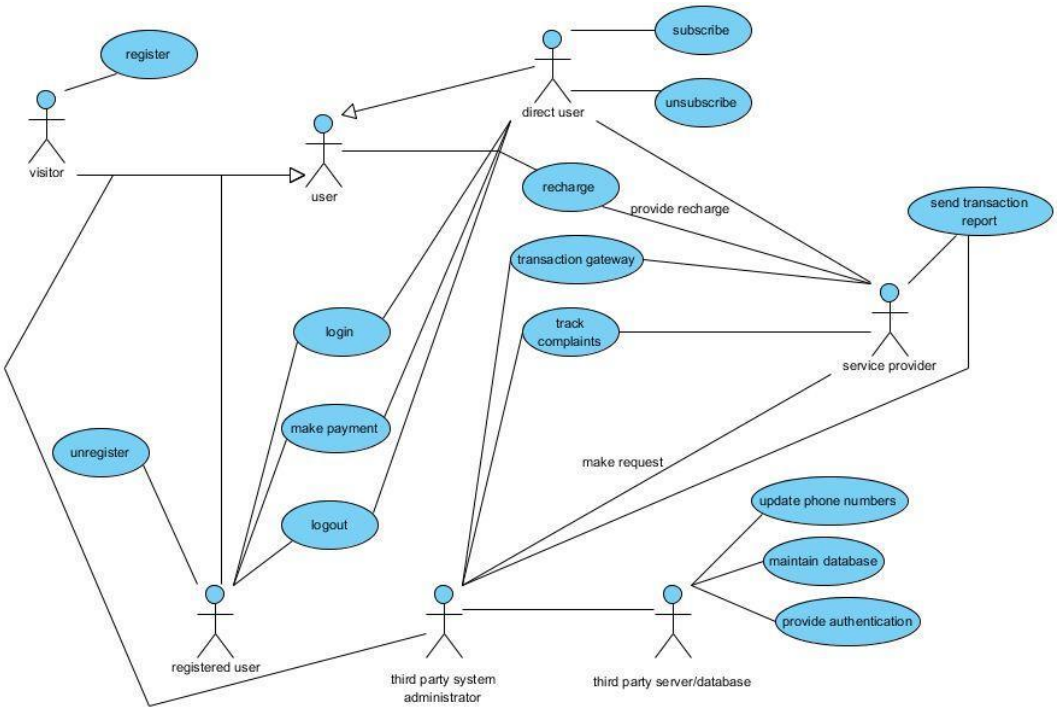


Fig 2.10 Use-Case Diagram

Sequence Diagram

Figure 2.11 shows the sequence diagram arrangement of mobile recharge system where user will able to recharge his account using the help of third party:

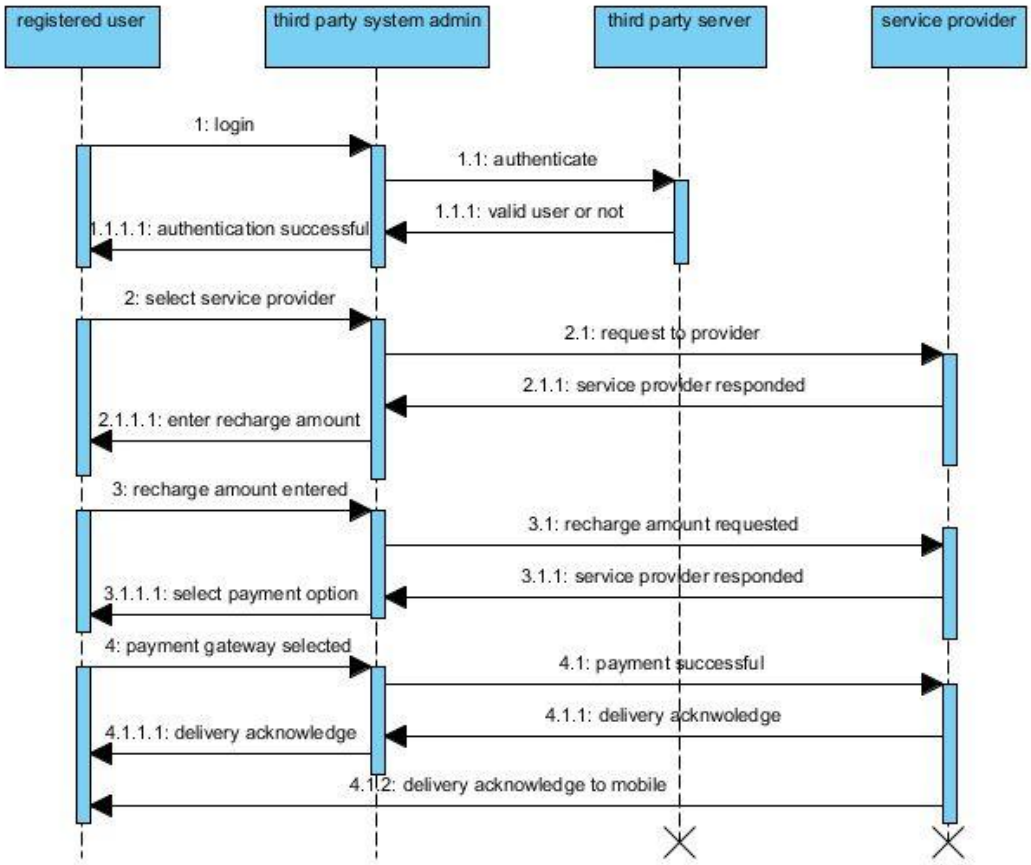


Fig 2.11 Sequence Diagram

Collaboration Diagram

Figure 2.12 shows the detail of collaboration diagram describing about mobile recharging by user that recharge his account through third arty site:

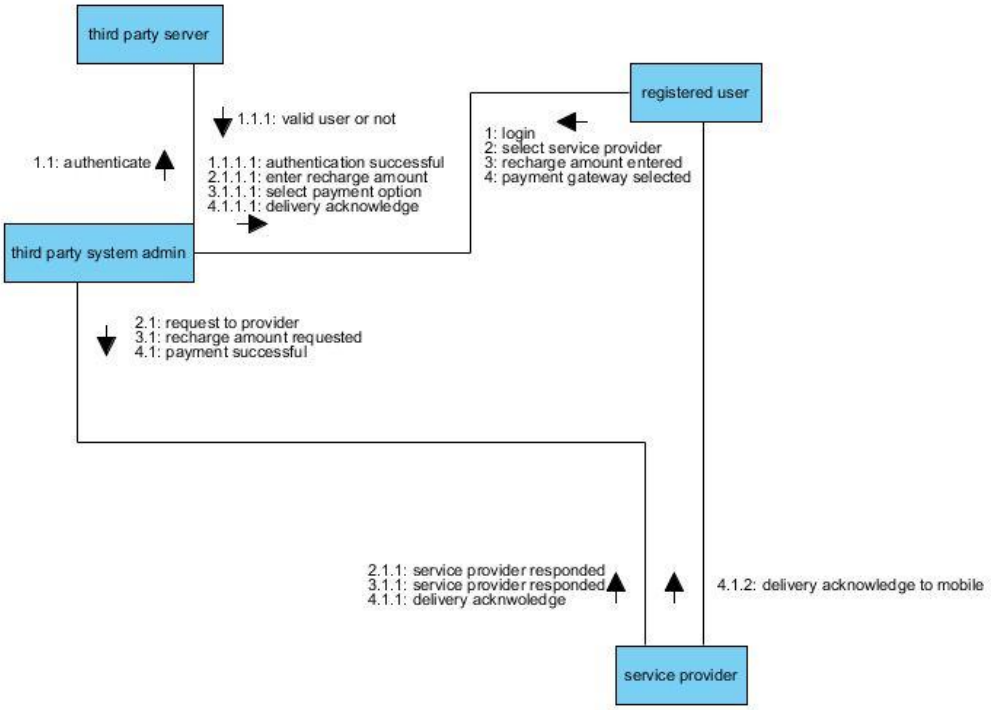


Fig 2.12 Collaboration Diagram

Activity Diagram

Activities:

- User login and authentication for Registered user.
- Forward the request to service provider if logged in as a Administrator.
- Enter service provider site for a direct user.
- Enter recharge amount.
- Select Payment Gateway.
- Login and authenticate Bank Account.
- Make payment.
- Check for the recharge processed successfully or not.

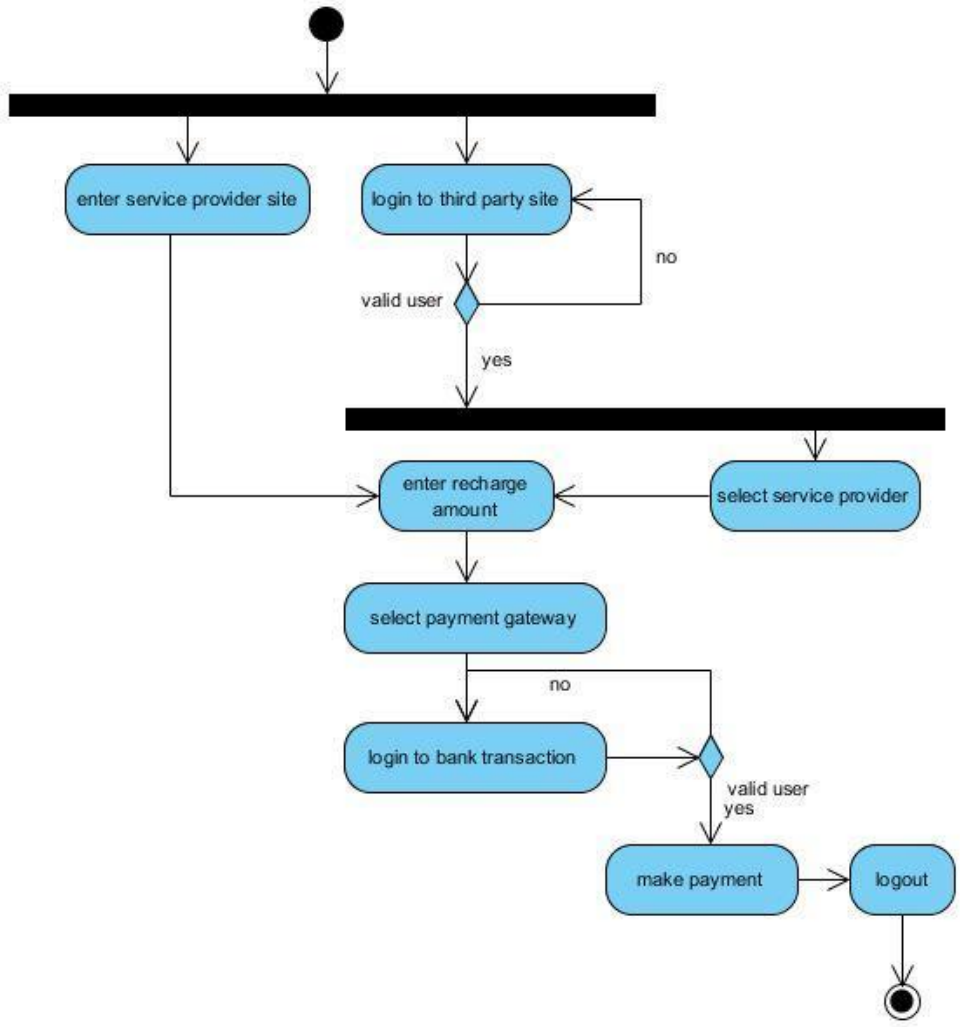


Fig 2.13 Activity Diagram

State Chart Diagram

States:

- Authentication for registered users / Registration for unregistered users
- Successfully logged on or re-login
- Operator Selection
- Show the tariff plans available and applicable
- Request recharge
- Go through Payment Gateway Transaction process

➢ Authentication to enter the gateway site

➢ Successfully logged on or re-login

➢ Payment made

Logged off

Transitions:

- Registration ---> Authenticate ---> Logged in
- Logged in ---> Operator Selection ---> Tariff Plans <---> Request Recharge
- Operator Selection ---> Request Recharge ---> Payment Gateway Transaction
- Payment Gateway Transaction ---> Operator Selection
- Payment Gateway Transaction ---> Logged off

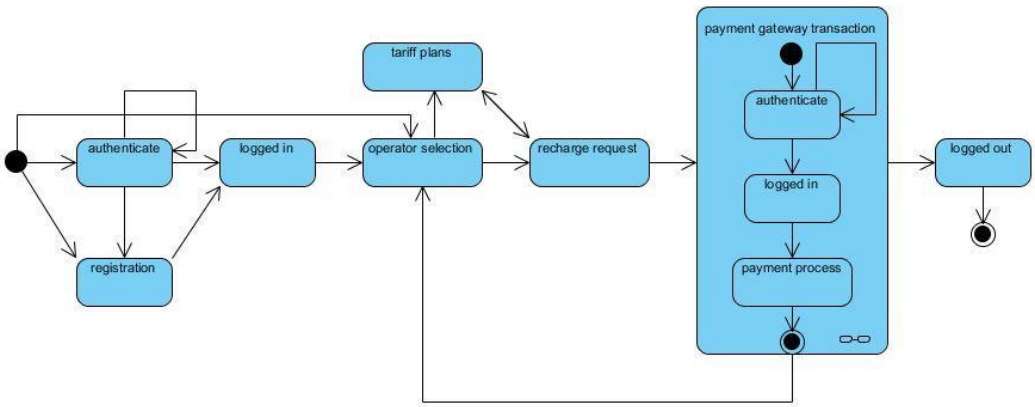


Fig 2.14 State Chart Diagram

Component Diagram

Components:

- Third Party Home Page (visitor / registered user / admin / service provider)
- Third Party Register Page (visitor)
- Third Party Login Page (registered user)
- Third Party User History Page (registered user)
- Request Recharge Page (registered user)
- Third Party Logout Page (registered user)
- Online Payment Transaction Gateway Page (direct user / registered user)
- Service Provider Home Page (visitor / registered user / admin / service provider)
- Tariff Plans Page (visitor / registered user / admin / service provider)

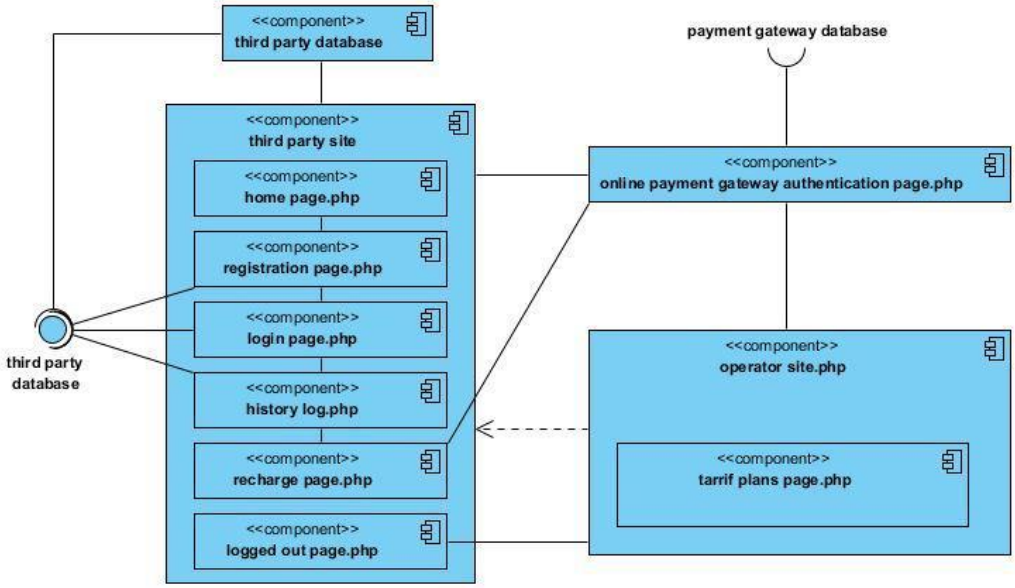


Fig 2.15 Component Diagram

Deployment Diagram

Systems Used:

- Consoles / Computers for registration, login purposes by third party users and for quick recharge by direct users.
- Third Party Server to receive and respond to all the requests from various users.
- Internet to provide access to users to recharge their accounts through payment gateways by placing requests through Third Party Sites and Service Providers sites.
- Payment Gateway Server like Bank's server to provide online payment through their personal accounts to meet the requirements of the users.
- Service Provider Server to maintain the records of the requests made by the users.

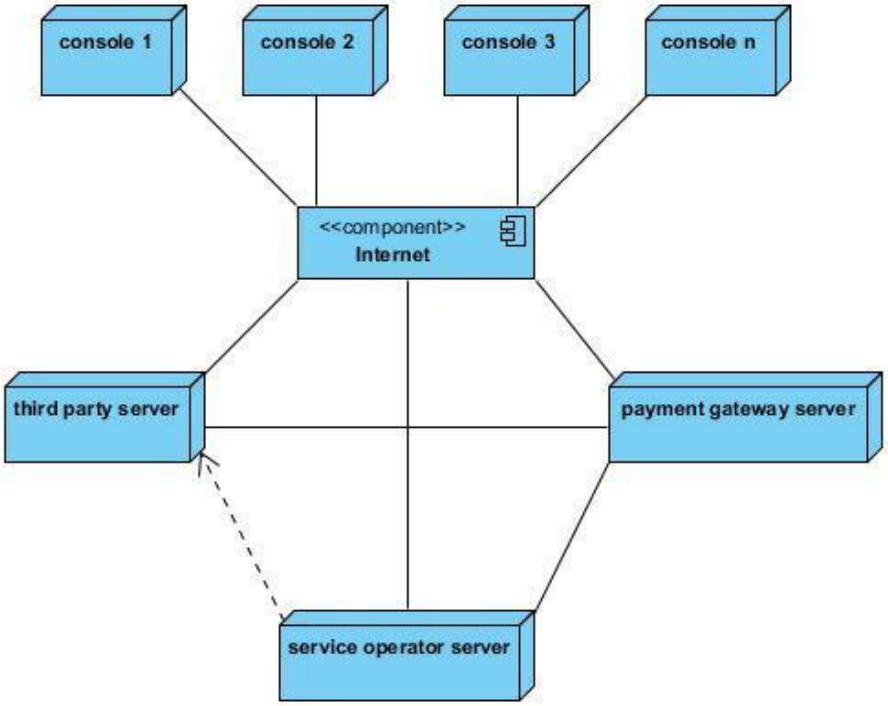


Fig 2.16 Deployment Diagram

Check your progress 2

1. Which of the following diagram is used to show the hardware and software of the system?

   a. Class diagram

   b. Deployment diagram

   c. Package diagram

   d. None of these

2.4 Let Us Sum Up

In this unit we have learnt that UML contains 2 different meta models as state charts and activity diagrams having different characteristics for reactive systems, particularly concurrency and hierarchy.

A class diagram is an arguably foremost applied UML diagram type which serves as building block of object oriented answer showing classes in exceedingly system, attributes and class operations.

The object diagram is Instance diagrams which is similar to class diagrams that shows relationship among objects used in real world and are applied to describe about system appearance at particular time.

The activity diagrams are used to model the behaviours of a system which means during which these behaviours are connected in an overall flow of the system.

2.5 Answers for Check Your Progress

Check your progress 1

Answers: (1 – d)

Check your progress 2

Answers: (1 - b)

2.6 Glossary

1. **UML -** A type of meta models having different characteristics for reactive systems.
2. **Class diagram -** An arguably foremost UML diagram type serving as building block of objects oriented answer showing classes in system, attributes and class operations.
3. **Object diagram -** Similar to class diagrams showing relationship among objects in real world that describes about system appearance at particular time.
4. **Activity diagram -** The model showing behaviour of system where behaviours are connected in overall flow of system.

2.7 Assignment

Write short note on UML Diagrams.

2.8 Activities

Collect some information on Online Mobile Recharge.

2.9 Case Study

Generalised Problem Statement and discuss.

2.10 Further Readings

1. C. Schulte, J. Niere: Thinking in Object Structures: Teaching Modelling in Secondary Schools; in Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts, ECOOP, Malaga, Spain, 2002
2. Zündorf: Rigorous Object Oriented Software Development, Habilitation Thesis, University of Paderborn, 2001.

Block Summary

In this block, you have learnt and understand about the basic of Component which can be extracted from Legacy Software along with detailed description of its needs. The block gives an idea on the study and concept of various standards needed for component software. You have been well explained on the concepts of JavaBeans.

In this block, you will understand about the features of Software components along with its characteristics. The concept related to Online Mobile Recharge with detailed description of use case are shown in detailed. You will be demonstrated practically about object diagram which is same as class diagrams with relationship in real world.

Block Assignment

Short Answer Questions

1. What is UML Diagrams?
2. Explain the function of Component in Software?
3. Write note on JavaBeans?
4. Write short note on Component Extraction from Legacy Software?

Long Answer Questions

1. Write short notes on various diagrams involved in Online Mobile Recharge?
2. What are the requirements for Component Based Development?
3. What are the advantages of Software Components?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any Other Comments

……

……

……

……

……

……

……

……

……

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
'Jyotirmay Parisar', Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi,
Ahmedabad-382 481.