SAD Notes

ICT 4th Semester

by

Mukesh Singh

ICT 5th Batch 2073

Sukuna Multiple Campus

Sundarharaincha-12, Morang

Unit-1

Information System Development

1.1 Fundamentals of System Analysis and Design

System → A system is a collection of elements linked together to achieve a specific goal.

It is a collection of elements ~~the~~ or components that are organized for a common purpose.

Information System → It is a set of processes and procedures that transforms data into information and knowledge.

System Analysis → It is a method of figuring out the basic elements of a project and deciding how to combine them in the best way to solve a problem.

System Design → System design is the process of defining the architecture, modules, interfaces and data for a system to satisfy the specified requirements.

Importance of System Analysis and Design

- ~~Data is~~ Easy to develop and rapidly
- The data is reliable, durable and effective
- Efficiency and flexibility
- Better management; better controls
- High quality information system can be developed.
- Less Costly

1.2 Process of System Development, CMM level

System development process is a set of activities, methods, best practices, and automated tools that stakeholders used to develop and continuously improve information system and software.

Capability Maturity Model (CMM) level

CMM is a methodology used to develop and refine organization's software development process.

It consists of 5 levels of maturity

Level 1: Initial — Processes followed are adhoc and immature and are not well defined.

- Unstable environment for software development.

Level 2: Repeatable -

- focuses on establishing basic project management policies

Level 3: Defined

- Documentation of standard guidelines & procedures takes place.

Level 4: Managed

Quantitative quality goals are set for the organization for software products as well as software process.

Level 5: Optimizing

- Highest level of process maturity
- focuses on continuous process improvement and process performance using feedback
- Use of new tools, techniques and evaluation of software process to prevent reoccurrence of known defects.

1.3

System life cycle vs Development

	System life cycle	System Development
1.	Is the overall life cycle of system from "cradle to grave".	It covers only the portion of time to develop the system.

Underlying Principles for System Development

Principle 1: Get the system users involved
- Get owners and users involved in all system development phases.

Principle 2: Use a problem solving approach
- Study and understand the problem in its context
- Define requirements of suitable solution
- Implement the solution

Principle 3: Establish phases and activities
- Scope definition - problem analysis - requirement analysis
- Logical design - decision analysis.

Principle 4: Document through the development
- Ongoing activity to reveal strength and weakness of the system during the development process

Principle 5: Establish standards for consistency
- System development standards
- Business standards
- IT standards

Principle 6: Manage the process and projects
- Process management and project management

Principle 7: Justify systems as Capital Investments

Principle 8: Don't be afraid to Cancel and revise scope
- Cancel the project if no longer feasible
- Reduce scope if budget/schedule is shrinking

Principle 9: Divide and Conquer
- Divide complex system into simple subsystems/components

Principle 10: Design Systems for Growth and Change
- Changes of technology, user requirements

System Development Life Cycle (SDLC)

It is a project management model that defines the stages involved in bringing a project from inception to completion.

Stages or phases of SDLC

1. Planning and selection - To find scope of problem and determine solutions.

2. Analysis and requirements - Understand of buisness needs and processing needs.

3. Design - Define solution system based on requirement and analysis design

4. Implementation - Construct, test, train users, install new system

5. Maintenance - Keep system healthy and improved.

1.4. Alternate Approaches to Development

Rapid Application Development (RAD)

RAD is a software development methodology that uses minimal planning in favour of rapid prototyping.

RAD means less talk more action.

Phases in RAD

i) Business modelling - information flow is identified

ii) Data modelling - define objects needed for business

iii) Process Modelling - data objects defined in data modelling are converted to achieve the business information flow.

iv) Application generation - Automated tools are used to convert prowen models into code and actual system

v) Testing and turnover - Test new components and all the interfaces

Advantages	Disadvantages
• Less development time	• Depends on strong team
• Increases reusability of components	• Requires highly skilled developers
• Encourages customer feedback	• High dependency on modelling skills

Agile Methodology

It refers to a group of software development methodologies based on iterative development, where requirements and solutions ~~are~~ evolve through collaboration between self-organizing cross-functional teams.

Agile Model	Waterfall Model
1. It proposes incremental and iterative approach to software design.	1 Development of software flows sequentially from start point to end point.
2 The design process is broken into individual models.	2 The design process is not broken into individual models.
3. Customers can see ~~product~~ delivery stages of the ~~project~~ ~~product~~ project.	3 Customers can only see the product at the end of project.
4. Less secure.	4- More secure
5. Testers and developers work together.	5. Testers work separately from developers.

Commercial of the Components (COTS)

COTS stands for Commercial of the Shelf. It describes software or hardware products that are ready made and available for sale to the public.

Eg - MS office is a COTS product.

Maintenance and Reengineering

* Maintenance

It is for the running the system till the age of the system. It stands for all the modifications and updations done after the delivery of the software product.

Types of maintenance

i) Corrective maintenance - Correct & fix problems
ii) Adaptive maintenance - keep software up to date
iii) Perfective maintenance - keep software usable over long period

* Reengineering

It makes the system new to work for another life span.

Stages of re-engineering

i) Reverse reengineering - discover principles of device throug analysis
ii) Code restructuring - analyze source code, update code documentation
iii) Data restructuring - requires changes in architectural or source-code
iv) forward reengineering - reimplement old function, design, add new ones, improve overall performance.

1.5 Automated Tools and Technologies

CASE Tools

CASE stands for Computer Aided Software Engineering. It means development and maintenance of software projects with the help of various automated software tools.

CASE tools are set of software application programs, which are used to automate SDLC activities, to develop software system.
Eg - Analysis tools, design tools, project management tools, documentation tools etc

Components of CASE Tools:
i) Central Repository - Central place of storage of useful information
ii) Upper CASE Tools - Used in planning, analysis & design stages of SDLC.
iii) Lower CASE Tools - Used in implementing, testing and maintenance
iv) Integrated CASE Tools - helpful in all stages of SDLC

Types of CASE Tools:
i) Diagram Tools - Used to represent system components, data & control flow among various software components.
Eg - flow chart maker tool.
ii) Process Modeling Tools - to choose a process model or modify it
Eg - EPF Composer
iii) Project Management tools - helps in storing & sharing project information.
Eg - Creative Pro Office
iv) Documentation Tools - help to gather requirements in diagrams
Eg - Accompa
v) Design Tools - help to design the block structure of the software
Eg - Animated Software Design
vi) Configuration management tools - It deals with
- version and revision management
- baseline configuration management
- Change Control management
Eg - Fossil, Git
vii) Change Control tools - deal with changes made to the software
viii) Programming tools - helps in building software product.
Eg - Cscope to search code in C

ix) Prototyping tools - helps to build rapid prototype
e.g - Serena prototype
x) Web Development tools - Assist in designing web pages
Eg - fontello
xi) Quality Assurance tools - to ensure confirmance of quality
Eg - Eg - Soapfest
xii) Maintenance tools - helps in maintenance phase of SDLC.
Eg - Hp quality center.

Application Development Environment
ADE is hardware, software and computing resources required for building software applications.

Unit-2

Feasibility Analysis

2.1. Feasibility analysis is the process of confirming that a strategy, plan or design is possible and makes sense.

Feasibility study

It is an assessment of the practicality of a proposed project or system.

four test of feasibity

1. Schedule feasibity - It ensures that project should be completed within given time constraint or schedule.
2. Technical feasibility - It determines whether the solution can be supported by existing technical or technology or not.
3. Operational feasibility - It determines whether the system and its components are operating effectively once it is developed.
4. Economic feasibility - It estimates the economic requirements of candidate system before investments funds are committed to proposal.

2.2 Cost-benefit Analysis Techniques

CBA is a technique used to compare total cost of programme/project with its benefit using a common metric.

It's a way to estimate the benefits and costs in the system development.

Following types of CBA techniques:

1. Payback Analysis - is a popular technique for determining how much time is needed before benefits overtake the costs needed.

Example

The management of Health Supplement Inc wants to reduce its labor cost by installing a new machine. Two types of machines are available in the market - machine X and

machine Y. Machine X would cost $18,000 whereas machine Y would cost $15,000. Both the machines can reduce annual labour cost by $3,000
Required- Which is the best machine to purchase according to payback method?
Soln:
Payback period of machine X = $18000 / $3,000
= 6 years
Payback period of machine Y = $15000 / $3000
= 5 years

According to the payback method, machine Y is more desirable than machine X because it has shorter payback period than machine X.

2. Return on Investment (ROI)

This analysis technique compares the lifetime profitability of alternative solutions.

The ROI of the solution is ~~the~~ a percentage rate that measures the relationship between the amount the busines get back from the invest and the amount invested.

$$\text{Lifetime ROI} = \frac{\text{Estimated lifetime benefits} - \text{Estimated lifetime Costs}}{\text{Estimated lifetime Cost}}$$

Example
Estimated lifetime benefit (ELB) = 22000
Estimated lifetime Cost (ELC) = 13000

$$\therefore \text{Lifetime ROI} = \frac{ELB - ELC}{ELC}$$

$$= \frac{22000 - 13000}{13000} = \frac{9000}{13000} = 0.692$$
$$= 70\%$$

3. Net Present Value (NPV)

It is a technique that compares the annual costs and benefits of alternative solutions.

We define the costs and benefits ~~of~~ for each years of the system's lifetime.

Example

NPV = PV(Benefit) - PV(Cost)

= 68136.91 - 100000

= -31,863.09

4. ~~Breakthrough~~ Break-even analysis

It is a technique widely used by production management and management accountants. It is based on categorizing production costs between those which are variable and those that are fixed.

2.3. Feasibility Analysis of Candidate System

We do a comparison for each of the candidate to choose which candidate is the best solution to be applied.

There are two can be used to make comparison.

1. Candidate system matrix - It is a tool to ~~make~~ compare the similarities and differences between candidate systems based on certain characteristic.

Example

Characteristics	Candidate 1	Candidate 2	Candidate 3
Portion of the system			
Benefits			
Software needed			
Input devices			
Output devices			

2 Feasibility Analysis Matrix

It is similar to candidate matrix system, but the different is it comes with an analysis and ranking of the candidate system. It has same columns as in candidate system matrix with an additional column named ranking column.

Example

	Weighting or ranking	Candidate 1	Candidate 2	Candidate 3
Description				
Operational feasibility				
Technical feasibility				
Economic feasibility				
Schedule feasibility				
Legal feasibility				
Cultural feasibility				
Weighted score				

Unit-3

Determining System Requirement

3.1 Requirement Discovery, System Requirements

Requirement Discovery

It is the process and tools used to identify system requirements of the users of the proposed system.

System Requirements

It is a description of the needs and desires for a system or application. It describes functions, features and constraints.

Two types of system requirements:

1. Functional Requirements - are functions or features that must be included in a system in order to satisfy the business needs and be acceptable to the system users.

Typical functional requirements include:

- Technical specifications
- System Parameters
- System Constraints
- Calculations
- Data manipulating and processing
- Business rules
- Administrative functions.
- Certification requirements
- Historical data

2. Non-functional requirements - is any requirement which specifies how the system performs a certain function.

OR, It describes how a system should behave and what limits are there on its functionality.

Specifies the system's quality, attributes or characteristics.

Typical non-functional requirements include:

- Efficiency
- Performance
- Services
- Information
- Control
- Scalability
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Security
- Regularity
- Usability

3.2 The Process of Requirement Discovery

~~1. Problem Discovery and Analysis~~

1. Problem Discovery and Analysis
- The requirement analyst should identify the problem along with the business users and define it accurately.

2. Requirement Discovery
It is the process and tools used to identify system requirements of the users of the proposed system.

3. Documenting and Analyzing Requirements
The requirements should be documented, clear, complete, consistent and actionable.

4. Requirements Management
Its purpose is to ensure that an organization documents, verifies, and meets the needs and expectations of its customers and stakeholders.

3.3 Traditional Methods for determining requirements

1. Interview - is one the primary ways to gather information about information system. A good system analyst must be good at interviewing and no project can be conducted without interview.
2. Questionnaire - have the advantage of gathering information from many people in a relatively short time. for effective survey, the analyst should group users properly and design different questionnaires for different groups.
3. Sampling - Predetermined no. of observations from large population
4. Survey - investigate the opinions or experience of a group of people by asking them questions.

3.4 Modern methods for determining requirements

1. Joint Application Design (JAD)

JAD is a faciliated, team-based approach for defining the requirements for new or modified information systems.

Primary purpose of using JAD is to analyst is to collect system requirements simultaneously from the key people involved with the system. to find conflicts.

2 Using Prototypes for Requirement determination

Prototyping is means of exploring ideas before you invest in them.

Prototyping allows system analysts quickly show users the basic requirements into a working version of the desired information system.

It is needed when user requirements are not clear or well understood.

3.5 Documenting Requirements using Use Case List

Use case is a list of actions or events steps typically defining the interactions between a role and a system to achieve a goal. The role (actor) can be a human or other external system.

It is a methodology used in system analysis to identify, clarify and organize system requirements.

Unit-4

Data Modeling

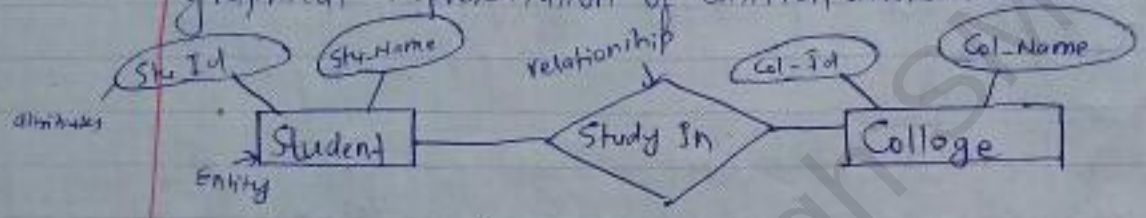
4.1

Data Modeling and analysis

Data modeling is a set of tools and techniques used to understand and analyze how an organization should collect, update and store data.

Introduction to ER Modeling

ER model also called an ER diagram, is a graphical representation of entities, attributes and relationships.

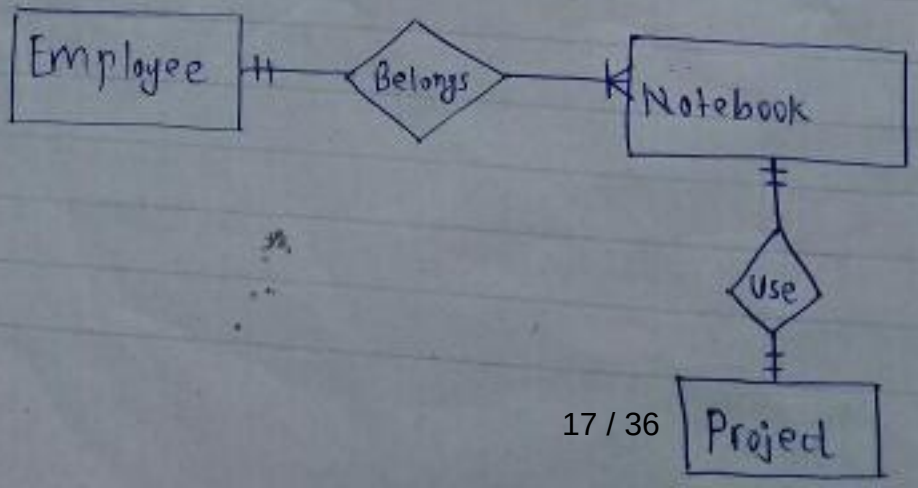


- plain and simple for designing
- It saves time
- It displays the clear picture of database

1) Entities - are definable things such as objects, persons or event.
2) a) Strong entity & b) Weak entity
2) Attributes - are properties or characteristics of entities
a) Simple & Composite b) Single & multivalued c) Derived attribute
3) Relationship - is bond or attachment between two or more entities.

Conceptual Data Modelling using ER Diagram.

A conceptual data model is a map of concepts and their relationships used for databases.



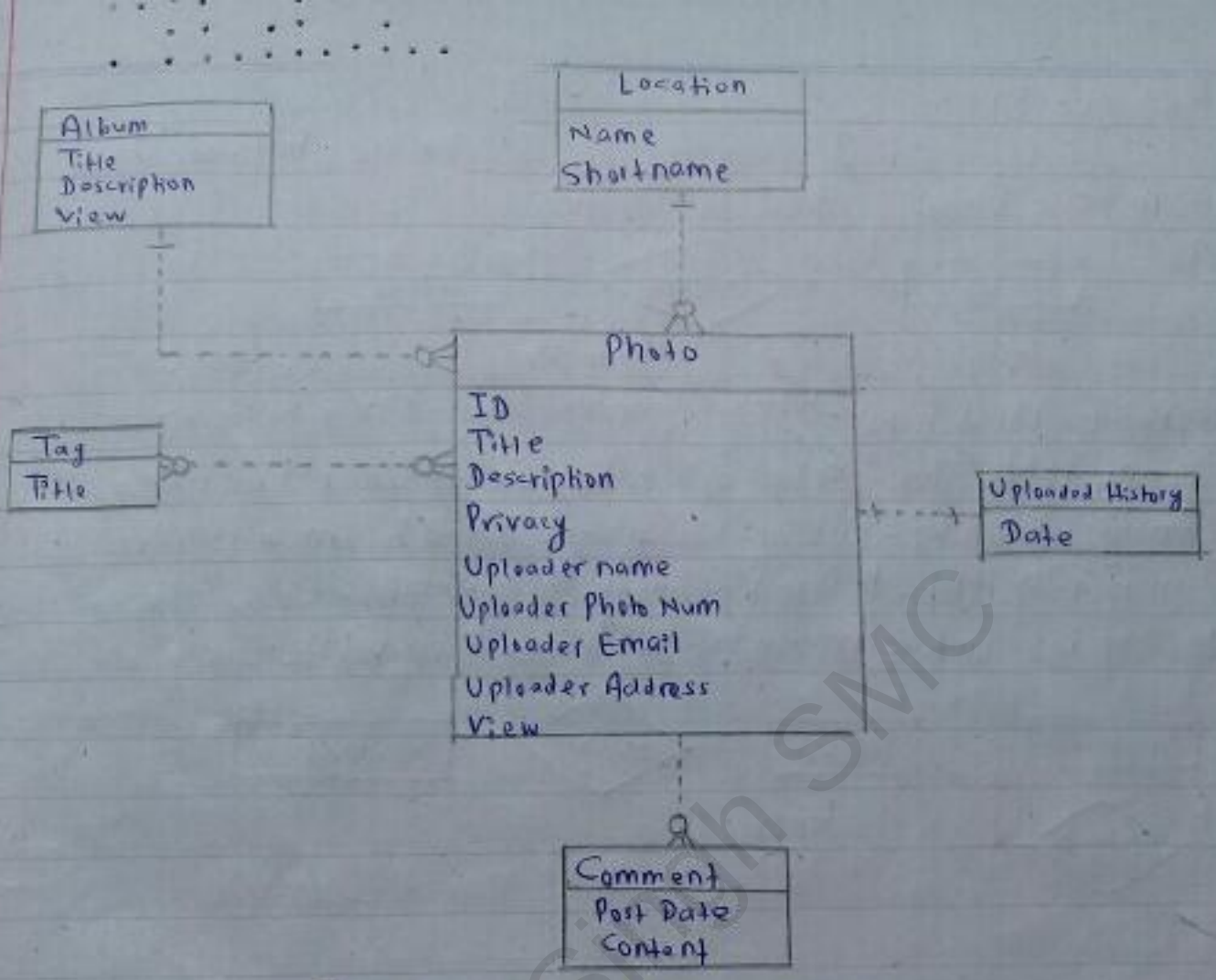
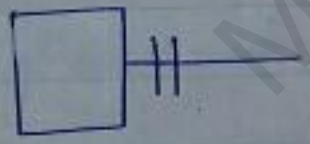
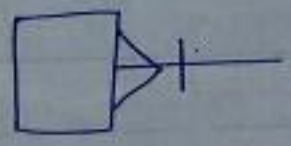


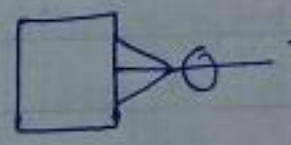
fig: Conceptual ERD example

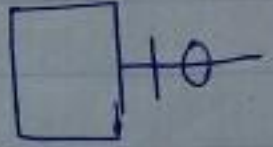
Crow's foot Notation of ER Diagram.

Crow foot Notation Symbols:



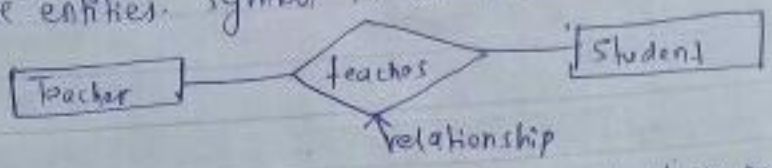






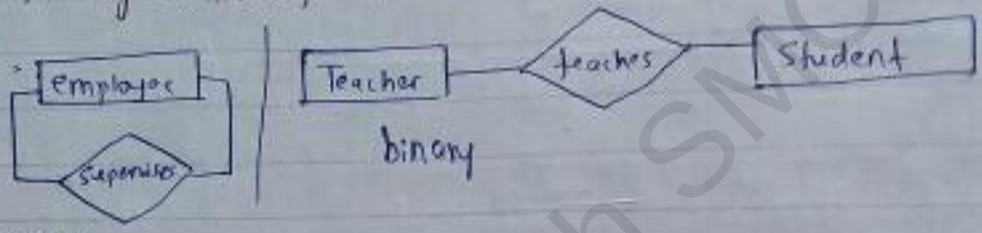
4.2 Relationships

It is defined as bond or attachment between two or more entities. Symbol is diamond



Degree of relationship - Number of entities involved in relationship

1. Unary relationship - only single entity involved in relationship
2. Binary relationship - two entities are associated to form a relation.
3. Ternary relationship - Relationship among three entities
4. N-ary relationship - n entities are involved in relationship

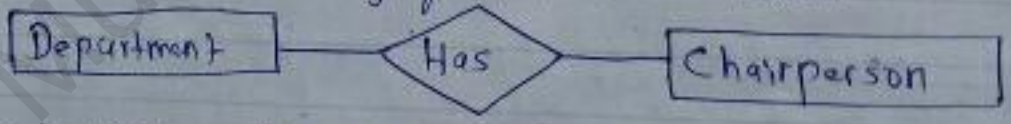


unary

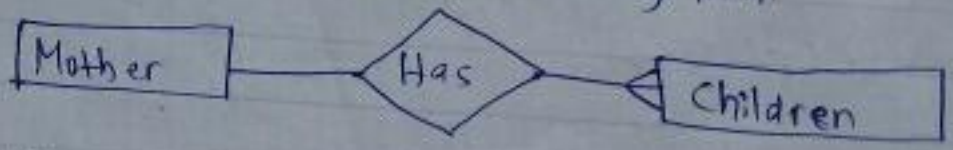
Cardinalities in Relationships

~~It defined~~ Cardinality of relationship is the number of entities in one entity set, which can be associated with entities of other set.

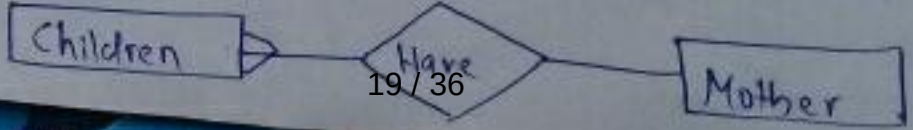
1. One-to-one → One entity from entity set A can be associated with at most one entity of set B & vice versa.



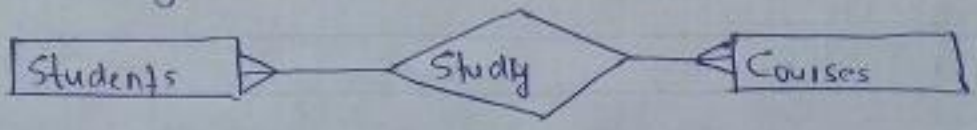
2. One-to-many → One entity from entity set A can be associated with more than one entities of set B but entity in B can be associated with at most one entity in A.



3. Many-to-one → One entity in set A can be associated with at most one entity in B but entity in B can be associated with one or more entities in A.



4. Many to many → One entity from A can be associated with more than one entity from B and vice versa.



Identifying relationship - The child table cannot be uniquely identified without the parent.

Example - Person Account (Account ID, PersonID, Balance)
Person (Person ID, Name)

Non-identifying relationship - Where the child can be identified independently of the parent.

Example: Account (Account ID, Account Num, AccountTypeID)
AccountType (AccountTypeID, Code, Name, Description)

Normalization

Process of converting complex data structure into simpler, stable data structure is called normalization.

Forms of normalization:

i) 1NF -
- Unique rows
- No multivalued attributes

Example

ID	Name	Course
1	A	C_1
1	A	C_2
2	E	C_3
3	M	C1
3	M	C2

The table is in 1NF as there is no multivalued attribute.

ii) 2NF

For a table to be in second normal form,

- It should be in 1NF.
- It should not have partial dependency.

Example:

Consider following functional dependencies in relation

R(A, B, C, D)

AB → C [A and B together determine C]

BC → D [B and C together determine D]

In the above relation, AB is the only candidate key and there is no partial dependency, i.e any proper subset of AB does not determine any non-prime attribute.

iii) 3NF

A table is said to be in third normal form when,

- It is in 2NF
- It does not have transitive dependency.

Example

Consider a relation R(A, B, C, D, E)

A → BC,

CD → E,

B → D

E → A.

All possible candidate keys in above relation are {A, B, C, D, E}. All attributes are on right sides of all functional dependencies are prime

Importance / Need of Normalization:

- To minimize data redundancy (duplicate data)
- To minimize or avoid data modification issues.
- To simplify queries.
- To reduce the need to reorganize data in

Unit - 5

Process Modelling

5.1.

Process Modelling: is a technique for organizing and documenting the structure and flow of data through a systems process.

Data Flow Diagram (DFD).

DFD is a graphical representation of flow of data in an information system.

Components of DFD

i) External entities - are the sources and destinations of information ~~system~~ entering or leaving the system. They are typically drawn on the edges of the diagram.

ii) Process :- It transforms the input data into output data. Circle stands for process that converts data into information.

iii) Data Stores :- ~~Rep~~ are files or repositories that hold information for later use such as Computer files or databases. It is represented by an open-ended box.

~~iv) Data flow -~~

iv) Data Flow - Represents the movement of data from one component to the other. It is represented by an arrow.

Data Flow Diagramming Rules:

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in DFD go to another process or a data store.

5.2 Decomposition of DFD:

* Context data flow diagram

DFD Level 0 is ~~called~~ also called a Context diagram. It is basic overview of the whole system or process being analyzed or modeled. It is the highest level of DFD containing only one process representing the entire system.

Eg: Grading system, Registration system etc

Functional Decomposition Diagram (FDD)

It is top-down representation of process or function. It is used at various stages of system development.

Level-1 DFD:- Process in diagram 0 can be exploded further to represent details of the processing activities.

Level-2 DFD :- If you one step deeper into parts of level 1. It may require more text to reach the necessary level of detail about the system's functioning.

Level-n DFD :-

Guidelines for drawing DFD:

- Naming conventions:
  - Processes: strong verbs
  - dataflows: nouns
  - datastores: nouns
  - external entities: nouns
- No more than 7-9 processes in each DFD.
- Data flows must begin, end or both begin & end with a process.
- Dataflows must not be split.
- Dataflows should not be a control signal.
- Loops are not allowed.
- A dataflow cannot be an input signal.
- Decisions and iterative controls are part of process description rather than data flows.
- If an external entity appears more than once on the same DFD, then a diagnol line is added to the north-west corner of the rectangle.
- Updates to datastores are represented in the textbox as double-ended rows.

Logical DFD	Physical DFD
1. It depicts how the business operates.	1. It depicts how the system will be implemented.
2. The processes represent the business activities.	2. The process represents the programs, program modules.
3. The datastores represent the collection of data regardless of how the data are stored.	3. The data stores represent the physical files and databases, manual files.
4. It's how business controls.	4. It shows controls for validating input data, for obtaining a record, for ensuring successful completion of a process.

Unit-6

System Implementation and Operation

System implementation is a process of ensuring that the information system is operational. It involves:
- Constructing a new system from scratch
- Constructing a new system from the existing one.

OR. It is the installation and delivery of the entire system into production.

6.1 System Construction and Implementation

1. The Construction Phase:
   - Development, installation and testing of system components.
   - Implement the interface between the new system and existing production systems.

Activities
i) Build and test networks (if necessary)
ii) Build and test databases
iii) Install and test new software packages (if necessary)
iv) Write and test new programs

2. The Implementation Phase:
i) Conduct system test
ii) Prepare Conversion plan
~~iii) Installation strategies for conversion plan~~
iii) Train users
iv) Install databases
v) Convert to the new system

3. Testing
i) Unit testing or program testing → It is the testing of an entire program.
ii) System Testing → Tests a completely integrated system to verify that it meets its requirements. It is the bringing together of all Programs that a system comprises for testing purpose.
iii) Regression testing → Is a tuting done to verify that a code change in the software does not impact/affect the existing functionality of the product.

6.2 System Operation and Support

System Operation:

System operation is day-to-day, week-to-week, month-to-month and year-to-year execution of an information system, business processes and application program.

System Support:

System support is the ongoing technical support for users, as well as the maintenance required to fix any errors, ommissions or new requirements that may arise.

Types of Operation Support System

1) Transaction processing systems

Record and process business transactions.

E.g: Sales processing, inventory systems, accounting system etc

2) Process Control System

Monitor and Control physical processes.

E.g:- Using sensors to monitor chemical process in a petroleum refinery.

3) Enterprise Collaboration System:

Enhance team and work group communication.

E.g: Email video conferencing.

Operating and Support System functions:

i) Resource detection
ii) Resource Organization
iii) Configuration
iv) Monitoring
v) Alert / Event management
vi) Analysis report

6.3 Program / System maintenance, System recovery, System enhancement

System maintenance

The process of monitoring, evaluating and modifying of existing information systems to make required or desirable improvements may be termed as System maintenance.

Types of system maintenance:

1. Corrective Maintenance → Processing or performance failures are repaired and corrected.
2. Adaptive Maintenance → Program functions are changed or replaced to satisfy the information needs of the user.
3. Perfective Maintenance → Adding new programs or modifying the existing programs to enhance the performance of the information system.
4. Preventive maintenance → Changes made to the system to reduce the chance of future system failure.

~~System Recovery:~~

System ~~maintenance~~ Enhancement

Adding features to the existing system, modifying the code to support changes in the user specification.

Unit-7

Object-Oriented Analysis and Design

~~Object Oriented Analysis~~

Object Oriented Analysis and Design (OOAD) is a popular technical approach for analyzing and designing an application system or busines by applying object oriented programming as well as using virtual modelling throughout the development life cycles to foster better stakeholder communication and product quality.

Benefits

- The ability to tackle more challenging problem.
- Improved communication among users, anlysts, designers and programmers.
- Reusability
- Increased Consistency

7.1 Object Oriented Development Life Cycle

1. Analysis Phase
   - Model of real word application is developed showing its important properties.
   - Model specifies the functional behaviour of the system.
2. Design Phase
   - Analysis model is refined and adopted to the environment.
   - The complete architecture of the desired system is designed.
3. Implementation Phase
   - Design is implemented using programming languages or database system.
   - The design model developed is translated into code in an appropriate programming language or software tools.
   - The code is then tested using specialized techniques to identify and remove the errors in the code.

Unified Modelling Language (UML)

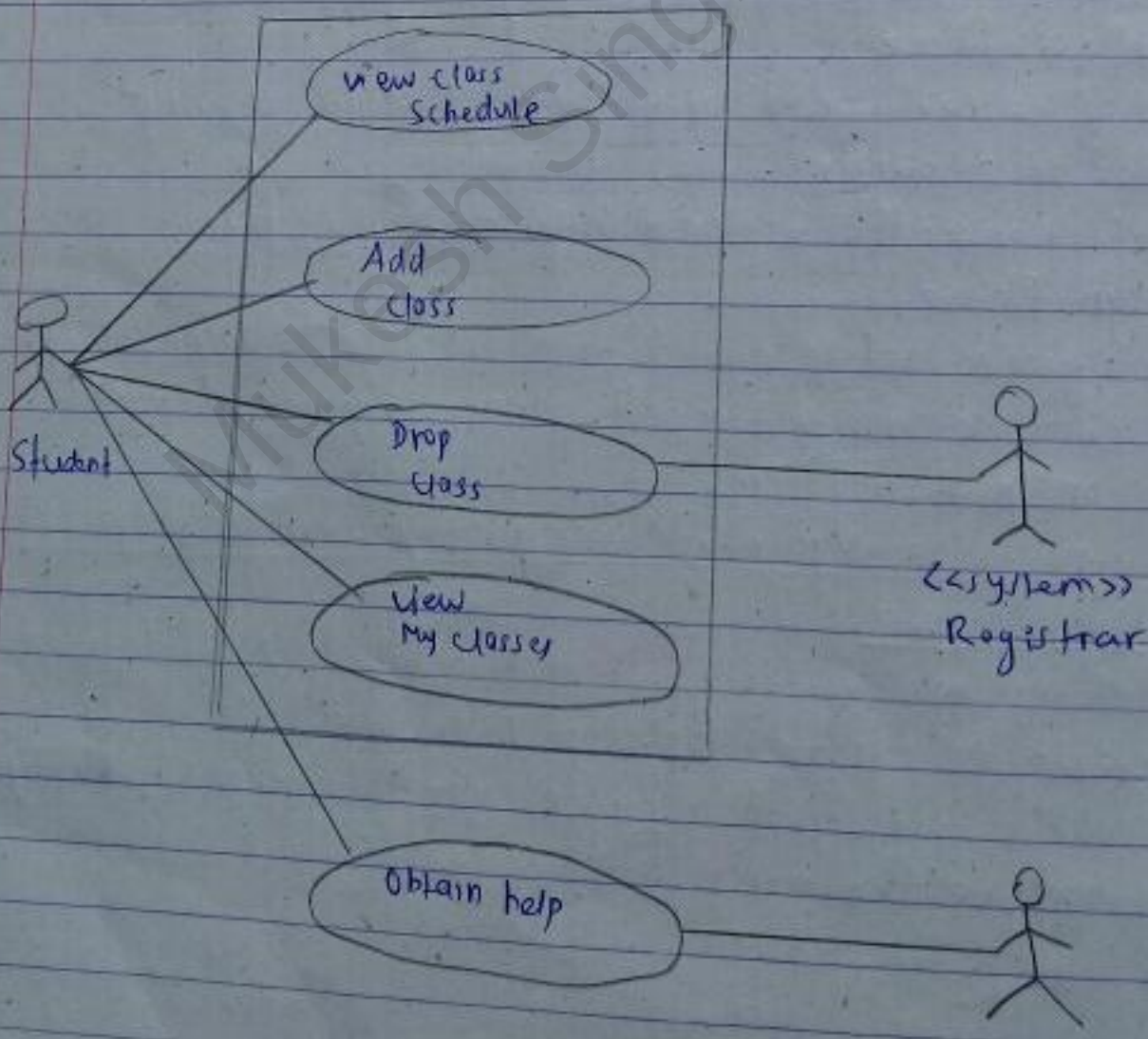
UML is a language for specifying, visualizing and constructing the artifacts of the software systems as well as for business modelling.

7.2. UML Diagrams:

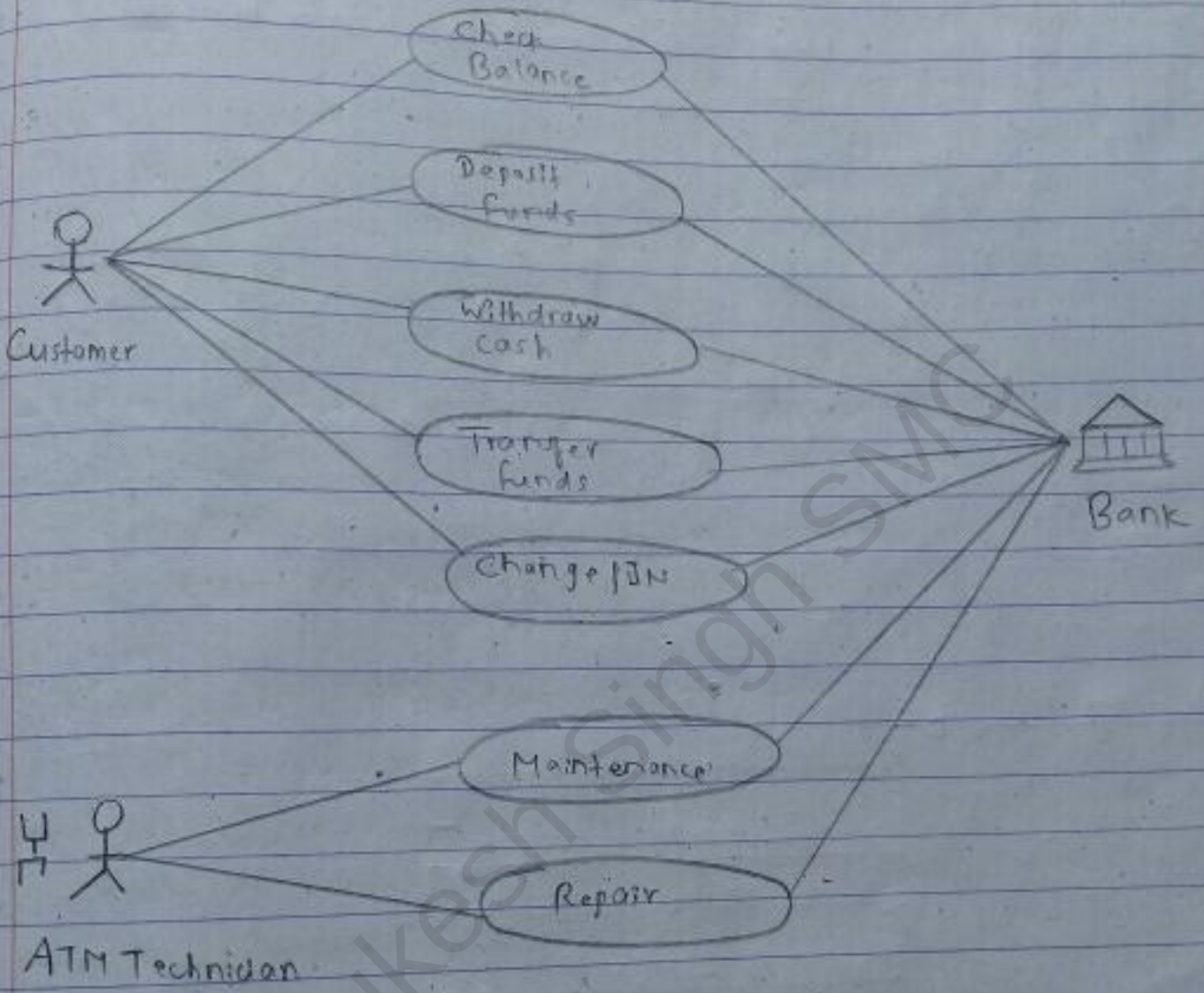
1. Use-Case Diagram
   - Applied to analyze functional requirements of the system.
   - Use cases represents complete functionality of the system.
   - ~~Use~~

Examples of Use Case Diagrams:

① Use Case Diagram of Class Registration



② Use Case Diagram for ATM System



2 Class Diagram

It is a type of static ~~diagram~~ structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations and the relationships ~~between~~ among objects.

→ Objects can participate in relationship with objects of the same class.

Essential elements of UML Class diagram are:

- Class name
- Attributes
- Operations

Example of class diagram:

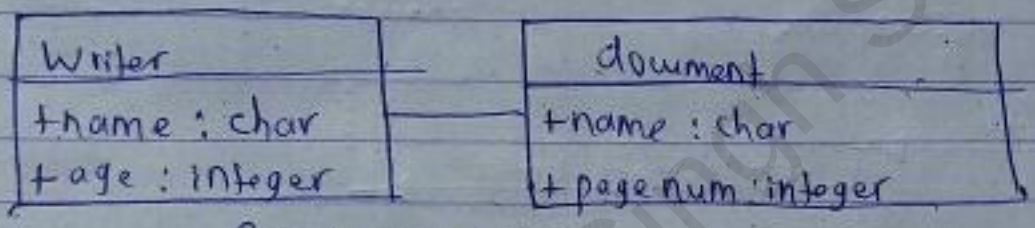


fig: Class diagram

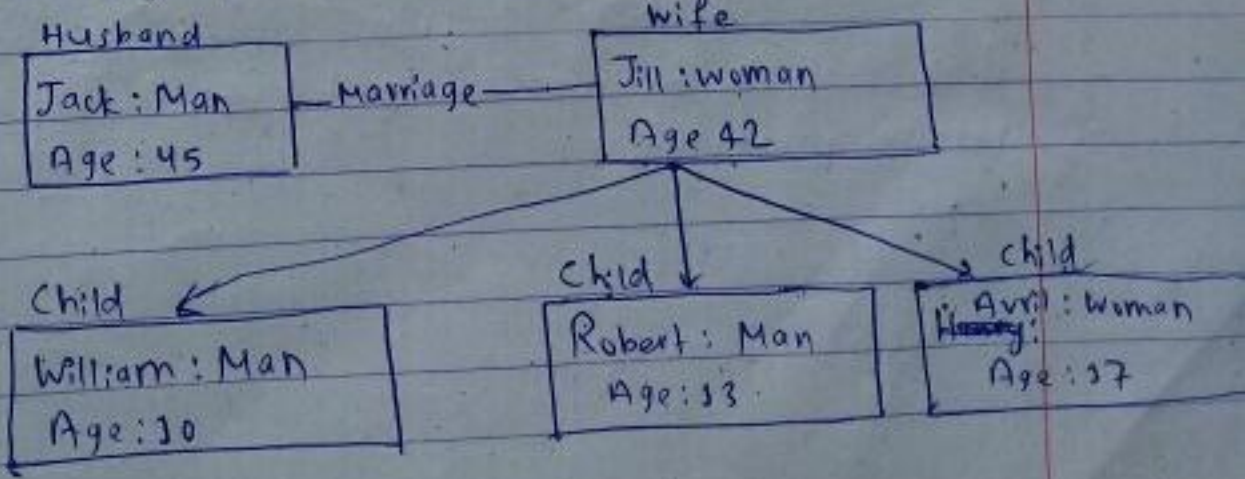
3. Object Diagram

- It is a graph of instances, including objects and data values.
- Object is represented as a rectangle with two compartments:

① Operation - A function or service that is provided by all the instances of a class.

② Encapsulation - The technique of hiding the internal implementation details of an object from its external ~~valve~~ view.

Example



Interaction Diagrams

Interaction diagrams are models that describe how a group of objects collaborate in some behaviour - typically a single use case.

It is of two types:

a) Sequence Diagram

A sequence diagram is a type of interaction diagram which describes how and in what order a group of object works together.

Used to capture the order of messages flowing from one object to another.

Example

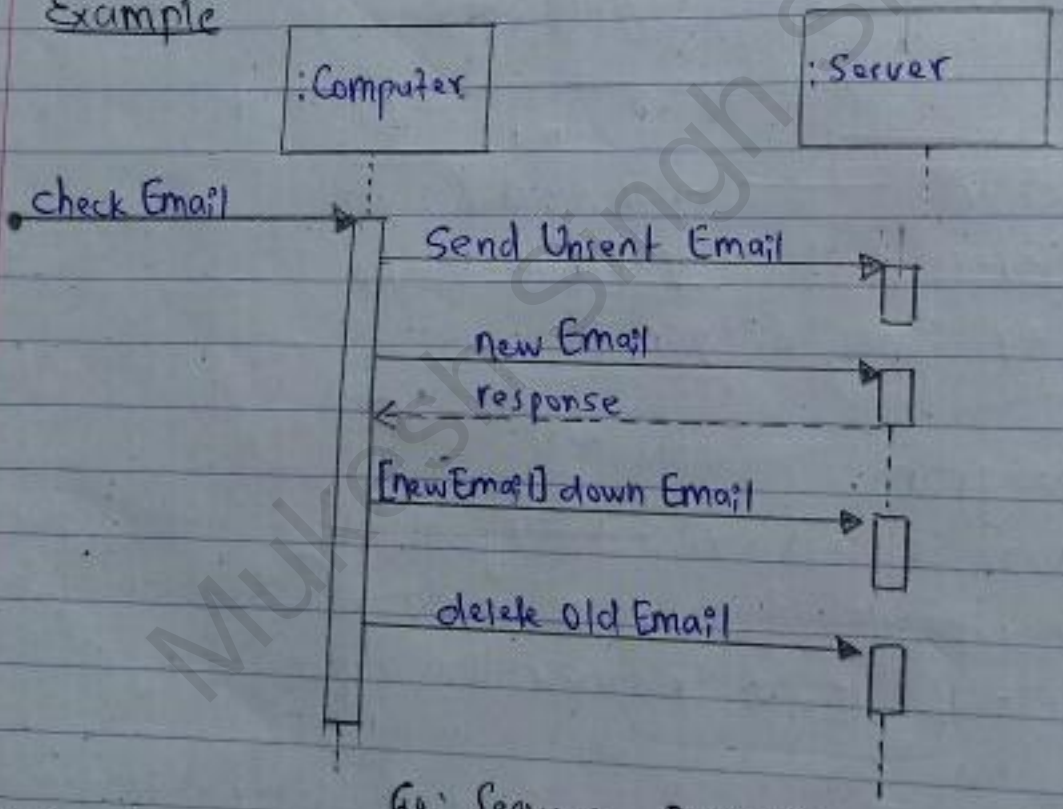


Fig: Sequence Diagram

b) Collaboration Diagram

Collaboration diagrams are interaction diagrams that illustrate the structure of the object that send and receive messages.

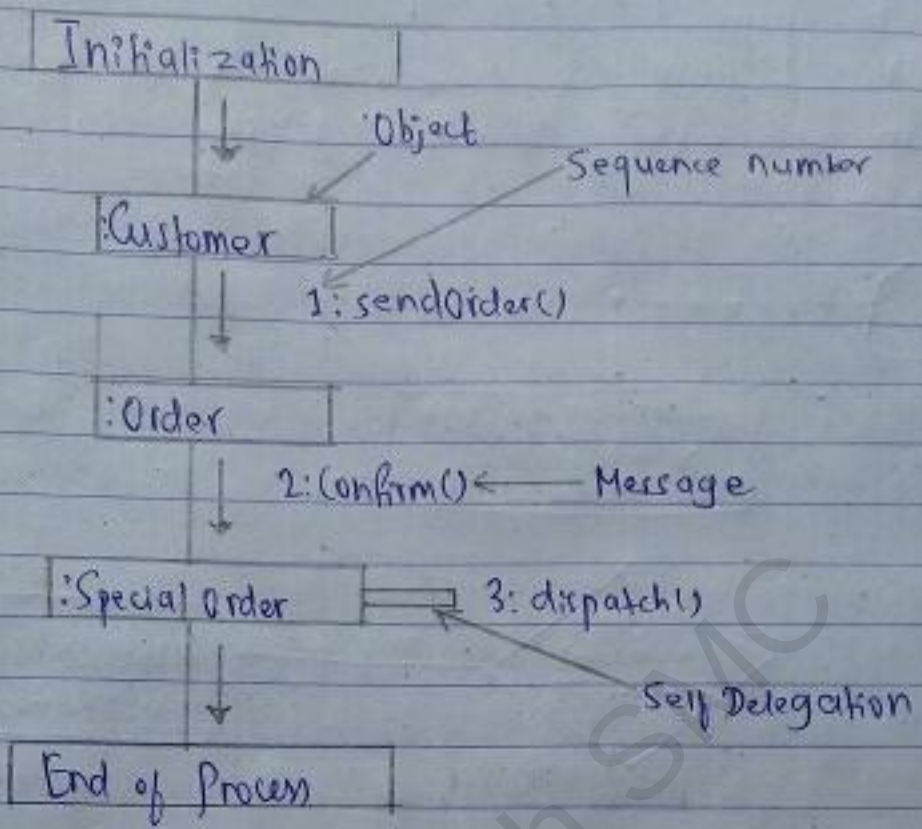


fig: Collaboration Diagram

State Diagram

It is a type of behavioral diagram in the Unified Modelling Language (UML) that shows transitions between various objects.

Example

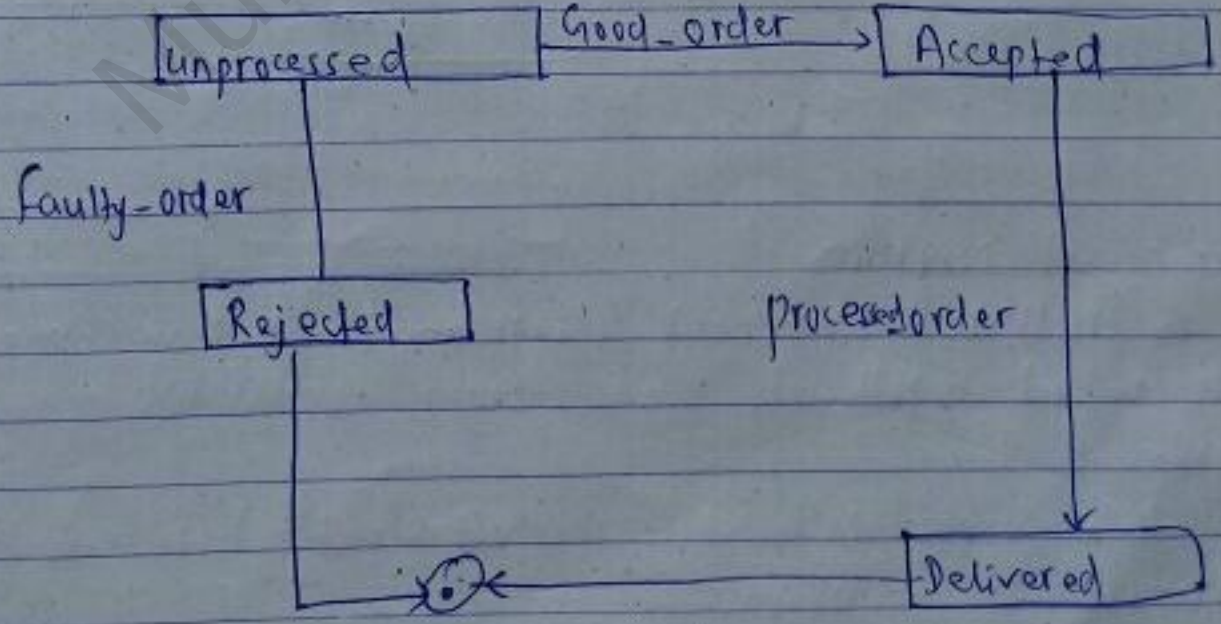
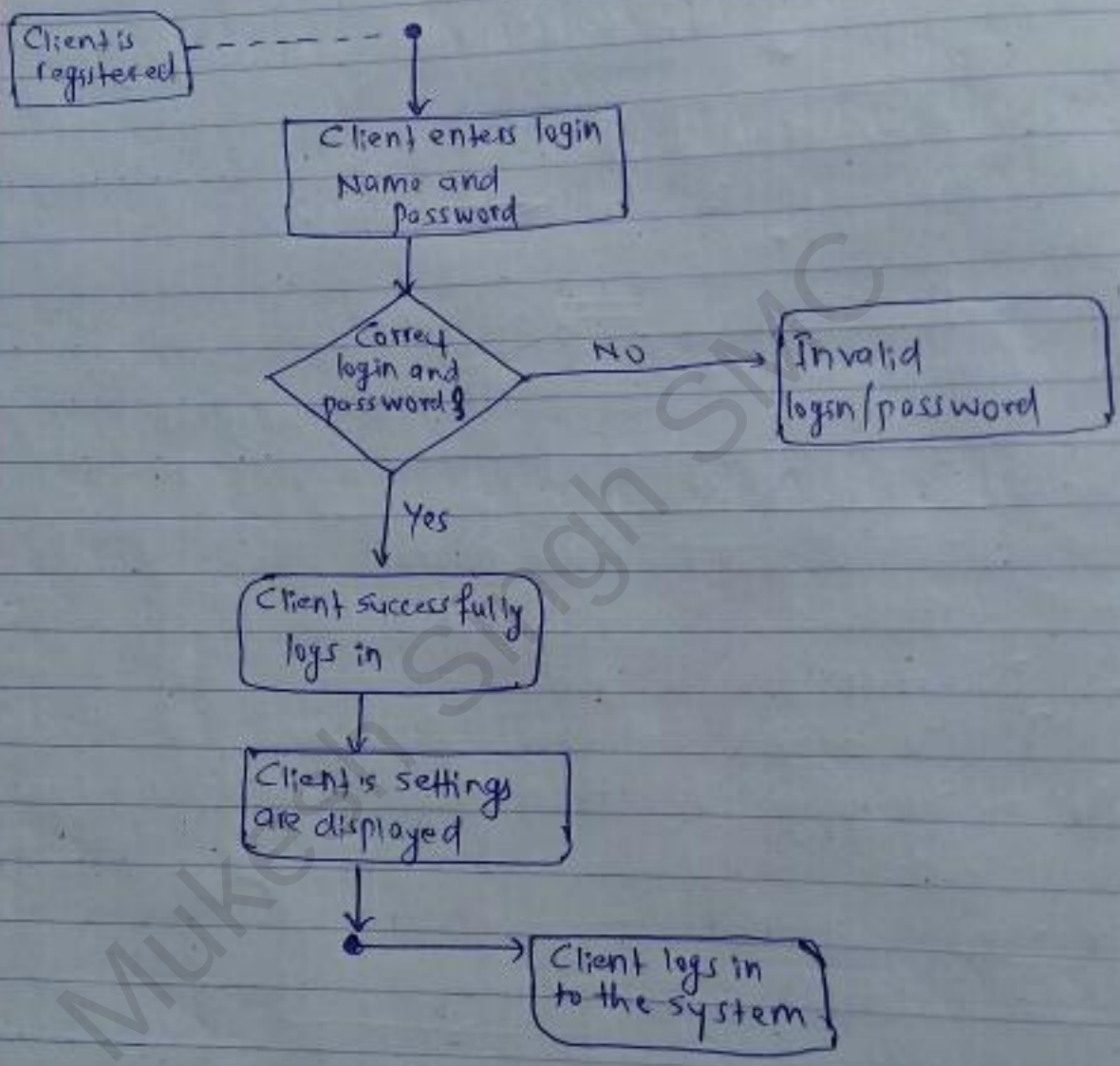


fig: State Diagram

Activity Diagram

It is basically a flowchart to represent the flow from one activity to another activity.

Example



Component Diagram

In UML, a component diagram depicts how components are wired together to form larger components or software systems.

It's purpose is to show the relationship between different components in a system.

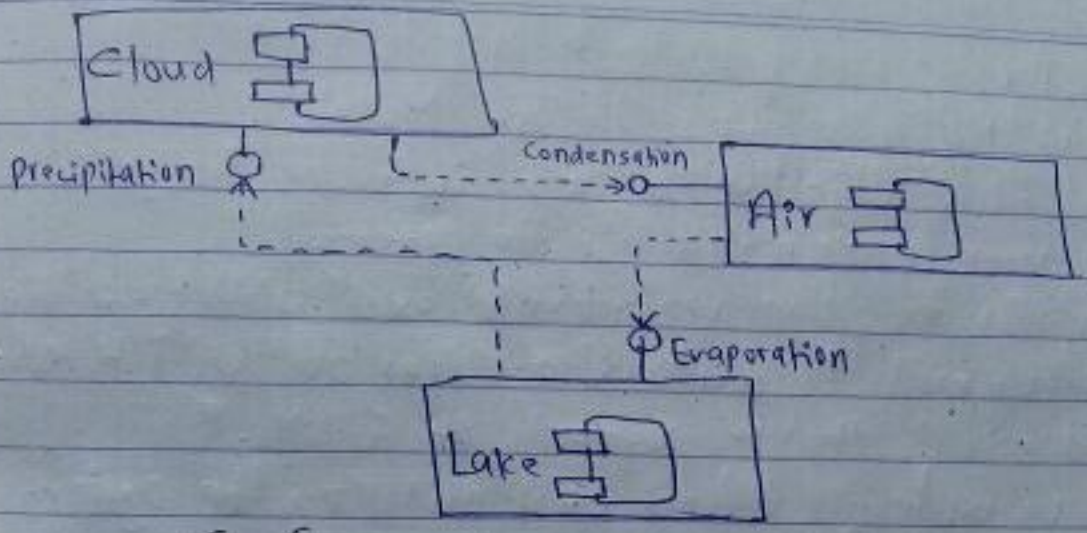
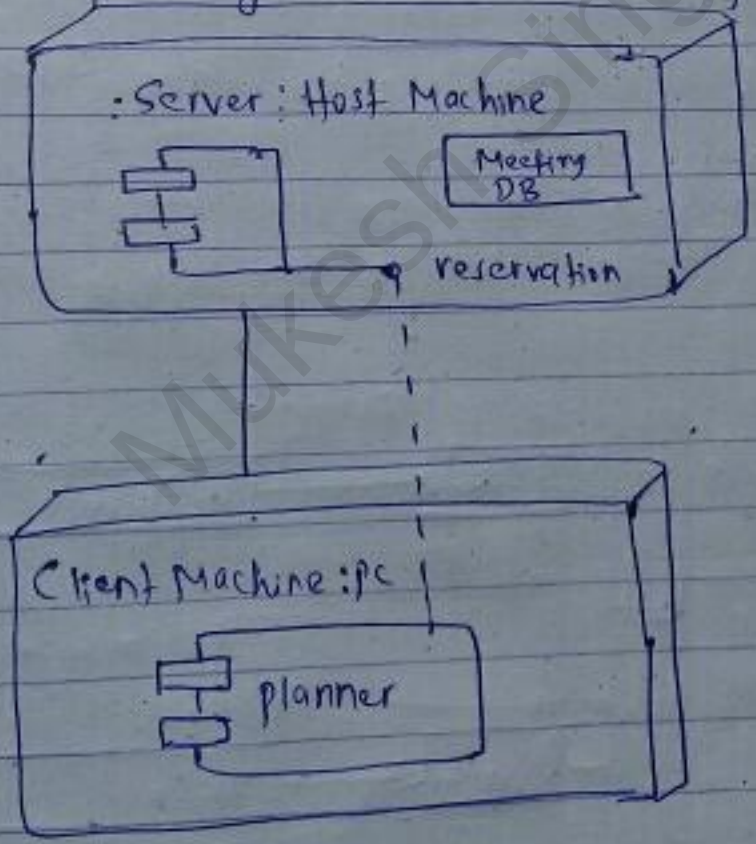


Fig: Component Diagram

Deployment Diagram

It is a diagram that shows the configuration of run time processing nodes and the components that live on them.



7.3. Object Oriented Analysis

Requirement Analysis using Use Case Model

The final objective of any software design is to satisfy the user requirements for the system.

Oftentimes ~~the~~ system requirements exist already in the form of requirement documents. Use cases are used to correlate every scenario to the requirements it fulfills. If the requirements does not exist, modelling the system through use cases enables discovery of requirement.

Conceptual Model

A Conceptual model is a model which is made of concepts and their relationships.

A conceptual model is drawn with a set of static structure where:

- Concepts are associated;
- Concepts have attributes
- Concepts have no operation

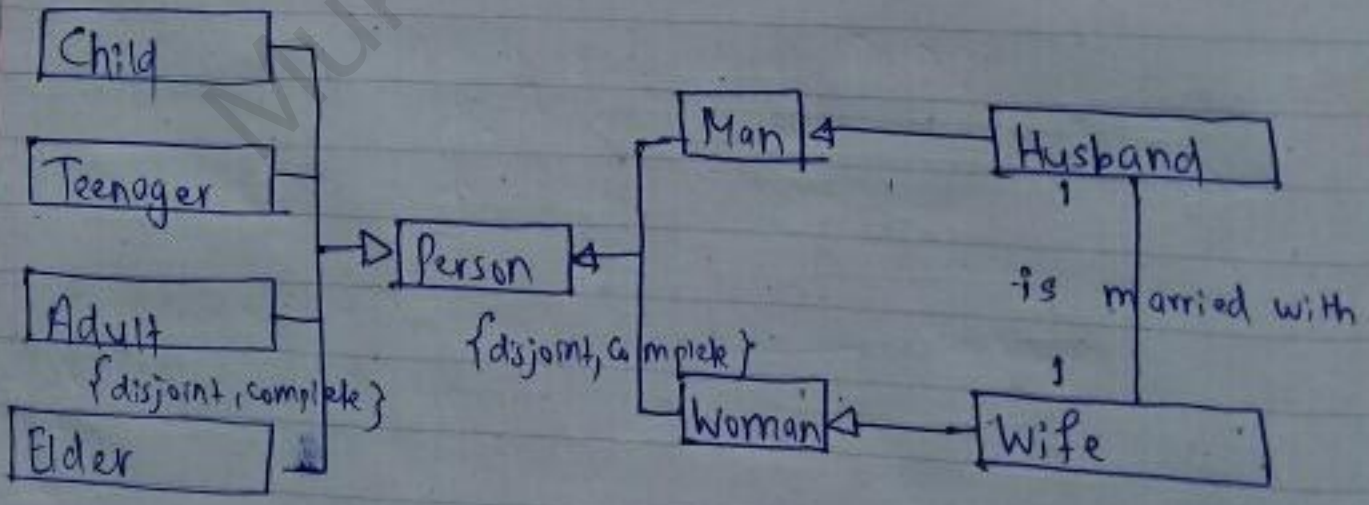


Fig: Structural Conceptual model example