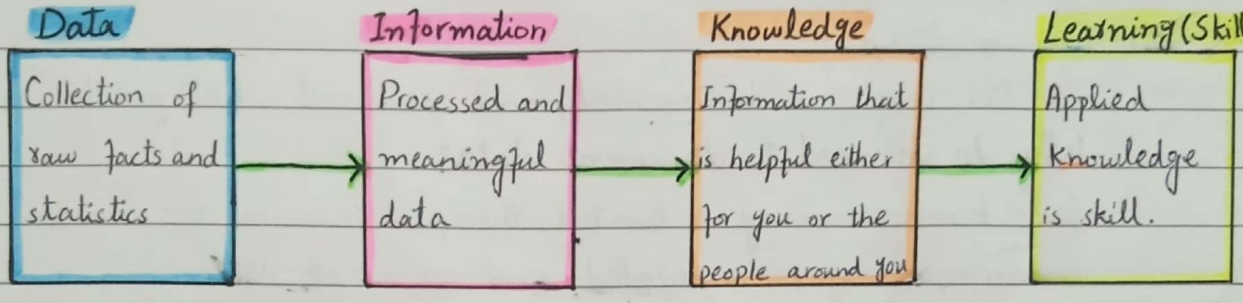
Lecture 01 Date: 12 June, 2023

Software Engineering

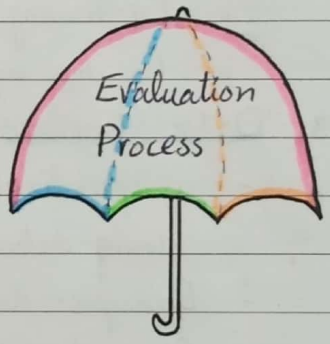
⇒ Evaluation Process



"The innate curiosity to explore the unknown makes an individual a true student."

Umbrella → Humbleness ↔ Gratitude

Umbrella of Evaluation Process can be achieved by humbleness. And humbleness is directly related to gratitude.

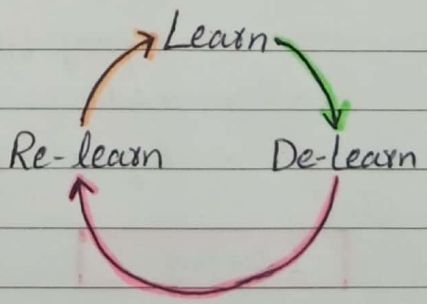


⇒ Learning Cycle

→ Learn
- previous knowledge/learning

→ De-learn
- forgetting previous knowledge

→ Re-learn
- getting new knowledge



Delearning is as important as relearning the new skills. In a new digital world, efficiency will still be important, but agility will be a top peroirity. To become more agile, each of us must de-learn traditional methods for how we work and what we do everyday.

Date: 14. June, 2023

Lecture 02

Software Engineering

⇒ Defining Software

1. Set of Instructions
2. Documentation

"Do what is documented, and document what you do."

Why do we need to document data?

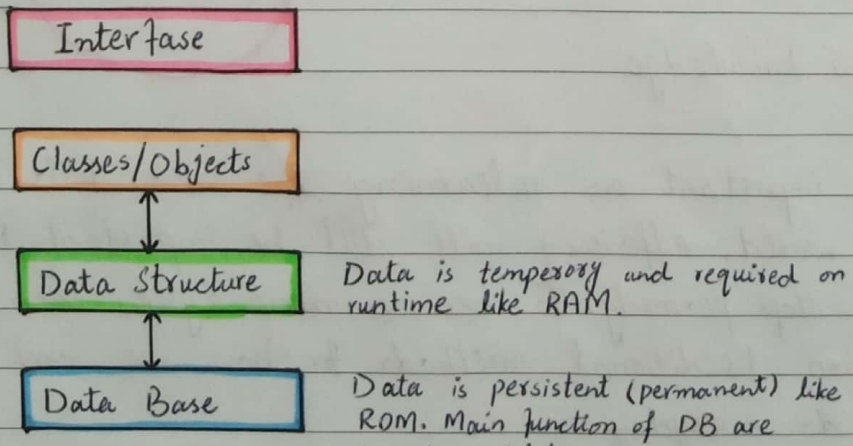
Since human memory is limited, therefore humans are unable to memorize large, complicated and sequential data.

Why do we need to maintain document?

i- For large size data
ii- To maintain data
iii- For having unity in thoughts of individuals
iv- For the improvement of data

3. Data Structure

- a specialized format for organizing, processing, retrieving and storing data.
- Example: Array, stack, tree, stats, graph, lists etc.
- Characteristics:
  - runtime (create on runtime)
  - temperory (flushed out when it's used)
  - volatile
  - They are vessels which are used according to requirement



Date:

1. Data
2. Function / Tasks / Activities / Process
3. Behaviour ↔ Attitude

"Behavior is driven by the attitude."

- The tip of iceberg is behavior and the remaining iceberg is attitude.

Date: 21 June, 2023

Lecture 03

Software Engineering

⇒ Why we make software?

- To produce easiness for users. (online banking, Daraz, Foodapp)
- To increase productivity. (less time consumed)
- To increase efficiency

⇒ Characteristics of Software

1. Software is engineered, not constructed.

Software	Hardware
- Intengible	- Tengible
- use mental energy to make	- use physical energy to maike

- SEI (Software Engineering Institute).
- DOS was the first version of MicroSoft. Unofficially it was launched in 1978, and officially in 1981.

2. Software doesn't wear out, but it does deteriorate.

When we make software, we need to figure out three things which are known as "Tripple Constraints".

i- Requirements (scope)

	Functional Req	Non-Functional Req
	- Basic requirements	- Additional or limited requirements
	- Define a system or its component.	- Define the quality attribute of the software system.
	- It specifies "what should the software system do?"	- "How should the software system fulfill the functional requirements?"
	- Specified by user.	- Specified by software developer.
Example:	- System shutdown in case of cyber attack.	- The processing of each request should be done within 10 seconds.

ii- ~~Cost~~ Time
- management
- in particular duration

iii- Cost

Date:

- These three constraints gathered to make "Quality" of software.
- Quality is the "satisfactation" of the customer. Quality doesn't mean excellence at each time.

3. Although industry is moving towards component-based construction, most software continues to be custom built.
   - Component-based Software
     - an approach to software development that relies on the reuse of entities called 'software components'.
     - Shortens production and delievy time.
   - Custom Built Software
     - software that is developed specifically for some specific organization or other use.
     - Increase production and delievry time.

Date: 18-Sep. 2023 Lecture 04

Software Engineering

=> Process and Software Process

→ Process

- A sequence of steps to achieve a particular goal.
- The understanding and clearance of goal determines the beauty of process execution.

→ Software Process

- In software process, two things are important; goal and step

○ What is the Goal of software Process?

- To achieve quality software
- substantial software (reasonable size and solution of problems)
- "Quality doesn't mean the excellence for each time for a person."
- Quality means customer satisfactation.
- Quality is not absolute, it varies.
- creeping the requirements (refers to how a project's requirements tend to increase over a project lifecycle). For example adding the debit card payment facility in payment method.

○ Steps to make Software

- In the production of any software, the process remains same.

Planning → Requirement gathering → Analysis → Design → coding → Testing → Debugging → Delivering → Maintenance

1. Planning
   - What to do? what is goal?
   - What are resources?
   - which sequence you required to do the tasks?
   - How much it costs?
   - How much time is required?
   - which task will be performed by whom?

- MS Project

"Failing to plan means plan to fail."

Date: _____

2. Requirement Gathering
   - Meeting is a technique to gather the requirements.
   - Interview is a technique to gather the requirements.
   - Questioning is a technique to gather the requirements.
3. Analysis
   - main focus on "what?". Try to understand and answer the what.
   - Structurally, in analysis, you need to understand three things.
     i- Data Modelling → ERD (Entity Relation Diagram)
     ii- Function Modelling → DFD (Data Flow Diagram)
     iii- Behaviour Modelling → STD (State Trassion Diagram)
     • Behaviour modelling deals data and function modelling simultaneously
4. Design
   - main focus on "How?"
   - Structurally, in design, there are four types of design
     i- Data Design
     ii- Architectural Design
     iii- Component Level Design
     iv- User Interface Design
5. Coding
   - provide syntax in programming language
6. Testing
   - Testing is done by ITG (Independent Testing Group)
   - Just identify the error. Don't correct the error.
7. Debugging
   - Error removing by coder/developer that are identified by ITG.
8. Delivery
   - when ITG clarifies 0 error, then we deliever the software.
9. Maintenance
   - There are four types of maintenance.

i- Correction
ii- Enhancement or Addition
iii- Adaptation
iv- Prevention

Date: 20 Sep, 2023 Lecture 05

Software Engineering

⇒ Environment during Developing a Software

→ Software Engineering: A Layered Technology

- **Quality Focus:**
  - Goal (Requirements/ Scope)
  - Time, Costs } (Resources)
  - Due to variation in these three combination/constraints, every software is different of the other.
  - Project must be unique.
  - This layer is also known as **Bed Rock**. Because on this, the development of a software is based on.
- **Process: (Glue)**
  - Process Model (extension of process)
  - The process used for particular situation is called process model.
  - For example: Giving tailer a dress before one day of eid.

  Situations and Process Models
  1. When the requirements are clear. (Linear Sequential Model)
  2. When the requirements are not clear. (Prototyping Model)
  3. When the time is short. (Rapid Application Development Model)
  4. When the risks are high. (Spiral Model)
  5. When you need to develop a commercial Software. (Incremental Model)
- **Methods:** (How to's)
  - How to accomplish each step of process.
  - The method is different to accomplish every task.
  - For example the methods to Analysis and Design (steps of process) are different.
- **Tools:**
  - supporting layer
  - also known as CASE (Computer Aided Software Engineering)

Date: 25-Sep, 2023

Lecture 06

Software Engineering

- SDLC activities (A → D → C → T) are sequential activities.

→ **Umbrella Activities**

- Activities that don't have sequence.
- Throughout the software process, these activities are continuously executed, or executed when they needed. That's why, these activities are called need-based activities.
- Examples
  - Project Management (supervised activity)
  - Software quality Assurance (continuous activity)
  - Risk Management (Need-based activity)

☺ Non-fundamental requirements are those requirements which are applied as constraints or limitation on functional requirements.
Typical Examples:

1. Security
2. Performance
3. Efficiency
4. Scalability
5. Interoperatability
6. Consistency

SDLC

Analysis → Designing → Coding → Testing

↓

(SRS)
Software Requirements Specification

↓

list of F and N.F requirements

Software Process Model

1. Linear Sequential Model

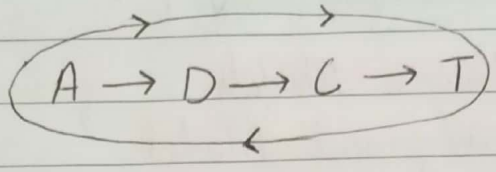
- Also known as waterfall model or SDLC
- classic life model (because it is first process)
- In this model, requirements of client are clear.

Problems:

- cannot accomodate the change
- don't allow the collection
- Unnecessary blockage

Date:

Iterative Development



2. Prototyping Model

- when the requirements are not clear
- reverse of linear sequential model. PM $\propto \frac{1}{LSM}$
- Look like, almost similar to the requirements of the client.
- Prototyping model is not only a process model, but it is a technique to gather requirements.

Problems:

- suitable for small or average software
- customer expect to deliever software soon because developer showed them the similar software. It increases the pressure on (clients) developers.
- Because of developer ↔ pressure can't make efficient software.

Date: 27-Sep, 2023 Lecture 07

Software Engineering

Software Process Model

3. Rapid Application Development Model

- When the time is short. (condition)

A. Increasement in Resources

- we should increase resources (experienced and skillful resourses) in this situation.
- This is pressure cooker situation, not normal situation.
- Agile Mindset (ready to accept the changes in situations, ready to face the problems and conflicts).
- Resources should be agile mindset.
- Resources should be used by parallel working.

B. Component Based Development

- There must be two catagories in components.
  - Independently function
  - Appropriate Interface (plug and play) (communication channel b/w two things)

C. General Management with Proactive Approach

- Proactive (ready for an activity before its execution)
- Proactive person always ready to update and open to learn, and is flexible.
- This model is lead by either project manager or team leader.

→ Steps to Make RAD Model

1. Business Modelling
   - In pictorial or diagram form
2. Data Modelling
3. Process Modelling
4. Application Generation
   - coding / implementation / construction
   - use 5th generation languages
   - Why we use 5th generation languages?
     Because in 5th generation, there are a lot of built

Date:

in functions. Hence, we don't need to write functions that are already available. This saves a lot of time.

5. Testing
   - There will be no normal testing
   - Automated tools are used to test.
   - Automated tools can't be alternative for manual testing.

4. Incremental Model

- This model is designed for commercial software.
- scope must be small for 1st version of product.
- requirements should be crystal clear
- work simultaneously on next updating versions of product.
- As you continue to give latest versions (increments), the previous versions should be stripped down.

Assignment #01

Evaluation of MS Windows.

Features: Added or changed.

Date: 2 Oct, 2023 Lecture 08

Software Engineering

Software Process Model

5. Spiral Model

- when the risks are high (situation)

→ **Risk:** (probability of loss)

Range of probability = 0.1 - 0.99 or
0.1% - 99%

There are different types of loss. Each loss can be showed in monitory terms.

severity of risk = (Probability of risk) x (loss)

→ **Types of Risk:**

- Human Resourse Risk
- Technology Risk } most common in software development
- Financial Risk }
- Social Risk / Religious Risk
- Infrastructural Risk

- Lack of information leads to high probability of risk.

→ **How to resolve or deal with risk?**

- change the situation in a particular way by either making it 0% or 100%
- Do the detailed homework (analyze data) and convert the probability into a certain situation.
- we do detailed homework at the start rather than at the time of execution.

→ **Steps to Make Spiral Model**

i- Determine Objectives (Tripple Constraints)
ii- Identify and Resolve Risk (Risk Management)
iii- Development and Test (SDLC)
iv- Plan the next iteration (Re-evelaution)

- Execution will be done after removing the risk (by multiple iterations)
- Risk can be removed by doing detailed homework.

Date: _____

→ Limitations

1. Spiral model process is a lengthy process.
2. It increases required time and cost.

- Software Houses generally use hybrid process model (combination of two or more than two process models).

→ Sequence to write a Software Process Model

1. State facts
2. Assumption (if required)
   - No assumption can be against the stated facts.
3. Select Software Process Model
4. Reasons (3 to 4 in bullet form)
5. Project working.

Date: 4 Oct, 2023

Lecture 09

Software Engineering

⇒ Project Management 4P's

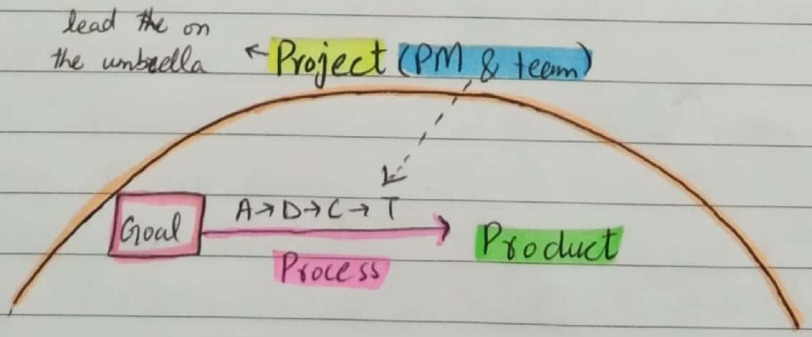
(characteristics) **Project** (first time)

(Uniqueness) - A project will be unique.
- Due to variation in tripple constraints, project will be unique.
- After hand over the project, when the project is start to being used, then it is no longer project, now it is operation and it is non-project.

(Non-Repitative)- A project is non-repitative. (Due to tripple constraints)
- But the operations will be repatative.

(Time Bound) - Project and non-project both will be time-bound.

(Goal Based) - Project and non-project both have goal but the goal is different. In case of project, the goal will depend on tripple constraints, but in non-project, the goal could be small and don't depend on tripple constraints.
- There will be some cases in which project and non-project overlap each other. eg. entry test (uniqueness everytime but procedure is repatative).
- The management of tripple constraints is called Project Management.



Umbrella of 4P's of Project Mangement

People

- Difference b/w management and leadership and their relationship
- A person can be manager and leader at the same time.
- Management is to keep things at right place and at right time.

Date: _____

- The foundation of the leadership is good management.
- MOI (Motivation, Organization, Innovation) model of leadership
- A leader has a vision and thinks out of the box.
- Creativity is just a new idea. Innovation is the practical of that idea.

Reading Assignment

John C. Maxwell

Article: Five levels of leadership

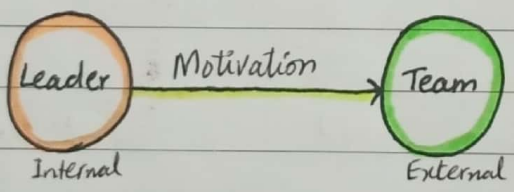
Date: 9 Oct, 2023 Lecture 10

Software Engineering

⇒ MOI Model

→ Motivation

- A leader must be able to motivate other.
- To keep yourself persuade to do a particular thing and stay focused on it is called motivation.
- Motivation varies from ~~one~~ point to point.
- 1st step to motivate others is motivate yourself (internal motivation).
- Demotivative person always think negatively.
- The nature of thinking depends on the experiences, thinking sources (reading, social media) and company (family, friends etc).
- The team that took motivation from leader, is called external motivation.



- Motivation, organization and innovation makes together a lethal combination.

⇒ Team Vs Group

Team	Group
- Team has objectives and goals.	- Groups have intersets, not goals and objectives.
- Members are bid highly interdependent.	- Members may work independently.
- Team have one leader.	- Group can have more than one leader or no leader

Date: _____

→ Tuckman's Team Model, 1965

1. **Forming:** Team is just formed. There is formality between team members and they treat each others like stranger.
2. **Storming:** Members start to communicate but still as individuals. They observe and judge other members and try to dominate on others.
3. **Norming:** Members act like team members. In this stage, they normalize things by compromising with each other and try to adjust.
4. **Performing:** (Long phase) Members work like team member according to the assigned tasks to each member.
5. **Adjouring:** When the assigned task is accomplished, contribution of each member is documented and the team is disabled.

→ Ground Rules

a) Communication
   - i- Vertical Communication
   - ii- Horizontal Communication

b) Decision Making
   - i- Consensus (100% agreeness)
   - ii- Majority (≥ 51%.)
   - iii- Plurality
   - iv- Order & Obey

c) Leadership
   - i- Parmanent
   - ii- Temporary

⇒ Team Structure

1. Control Centralized (CC) (Chief Programmer)
   - Communication: Vertical > Horizontal
   - Decision Making: Order & Obey (Dictatorship)
   - Leadership: Parmanent
2. Democratic Decentralized (DD) (Innovative Anarchy)
   - Communication: Horizontal > Vertical
   - Decision Making: With Consensus
   - Leadership: Rotation based
3. Controlled Decentralized (CD) (Agile Team)
   - Communication: Vertical ≥ Horizontal
   - Decision Making: as per situation
   - Leadership: Mix

Date: 11 Oct, 2023

Lecture 11

Software Engineering

Leader ⟷ Loader
↓ problem solver
↓ problem creator

Structured Analysis

There are three modellings in structured analysis.

1. Data Modelling
2. Functional Modelling
3. Behavior Modelling

1. Data Model : ERD

→ Why we make ERD?

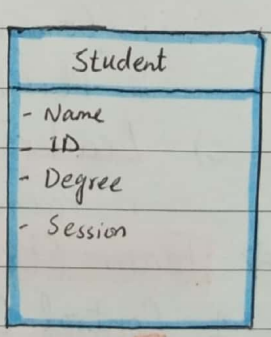
To understand the data domain of a system. (current system)

→ Elements of ERD

i. Entity (class)

- Anything in the universe which can be recognized by certain characteristics is called entity.

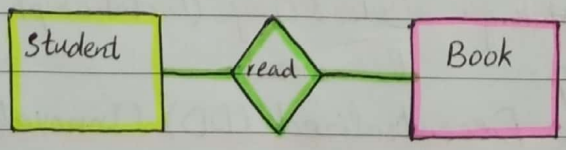
Entity ↓ instance ≅ Class ↓ object



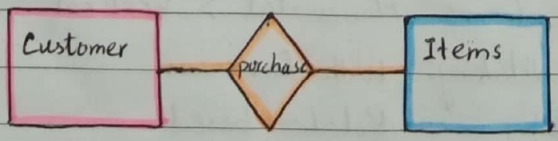
- Entity has no physical existance, but as we apply it, it will transform into physical existance called instance.

ii- Relationship

- connectedness / relatedness between two entities.



- Student reads book

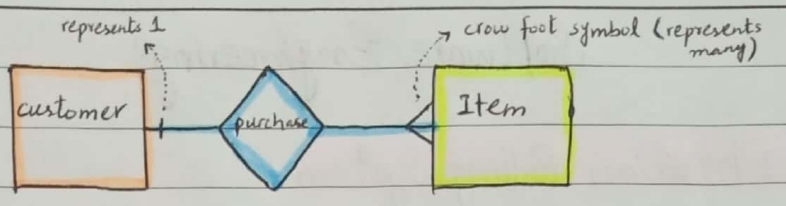


- customer purchases items.

- Relationship b/w entities is represented by diamond.

iii- Cardinality

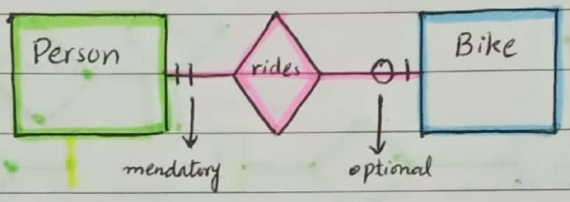
Date: _____



- When one instance of customer (entity) is created, it can create multiple instances of item. But when one instance of item is created, it will only create one instance of customer.

iv. Modality

- Modality indicates wether it is mendatory or optional to create an instance of entity against other entity's instance.

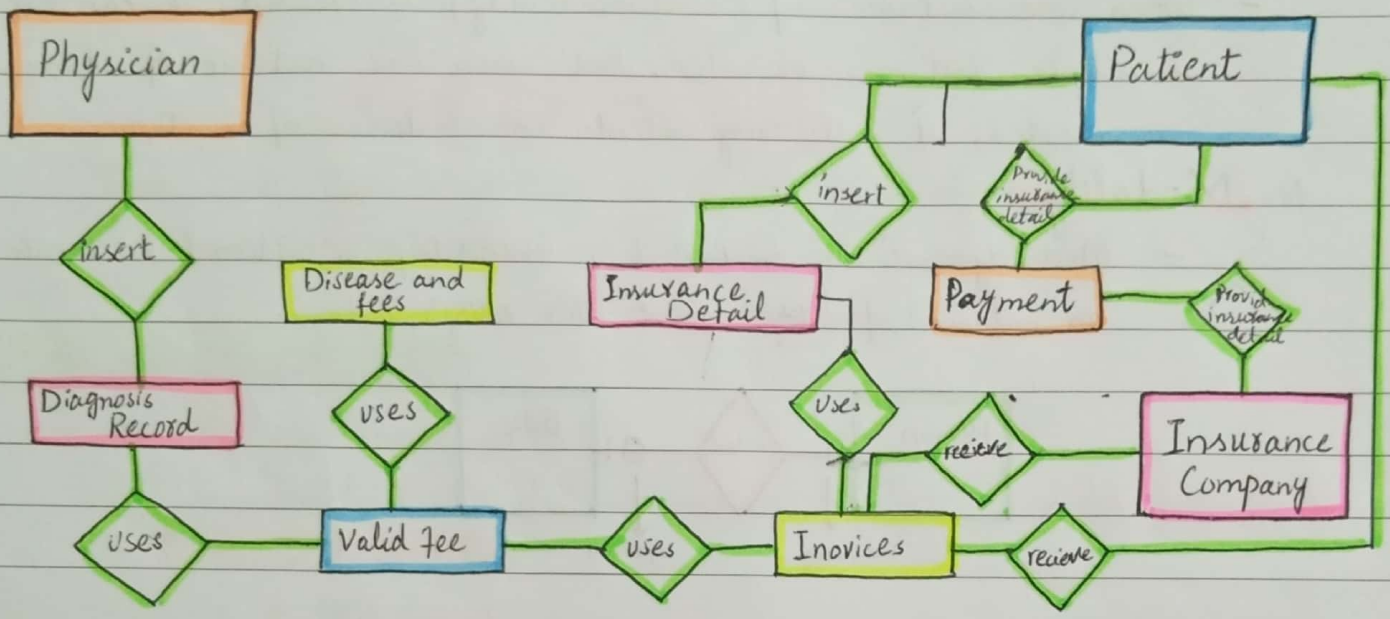


Date: 16 Oct, 2023

Lecture 12

Software Engineering

→ ERD of Physician Billing System



2. Functional Modelling: DFD

- DFD is a visual representation of the information flow through a process or system.
- Helps you understand process or system operations to discover potential problems, improve efficiency, and develop better process.

→ DFD Notation

1. External Entity

- Entity desired outside of the system.

i- All human beings are external entity.

ii- All exeternal systems are external entities.

- Any other system that interact with the given/current system is called external system. For example, PU with PUCIT, HEC with all Universties.

2. Process

- It is a function inside system which takes input and do some work on input and produces some intended output.
- Example: To login, to select items, to generate invoice, to generate report
- It is represented by (an arrow) a circle.

Date:

3. Data Store

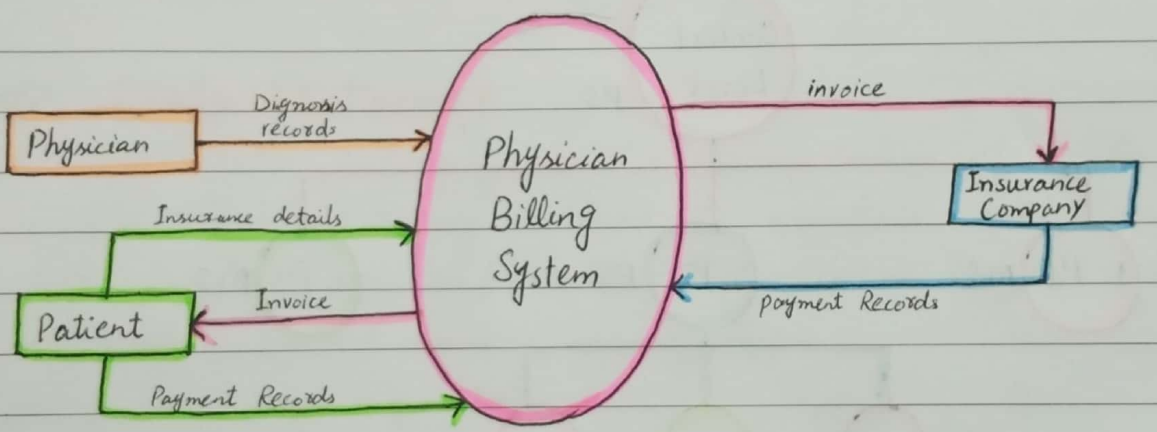
- It is a container which stores data permanently inside the system.
- Examples: Student data in CMS, subject data in CMS, User data for ver management etc.

4. Data Flow

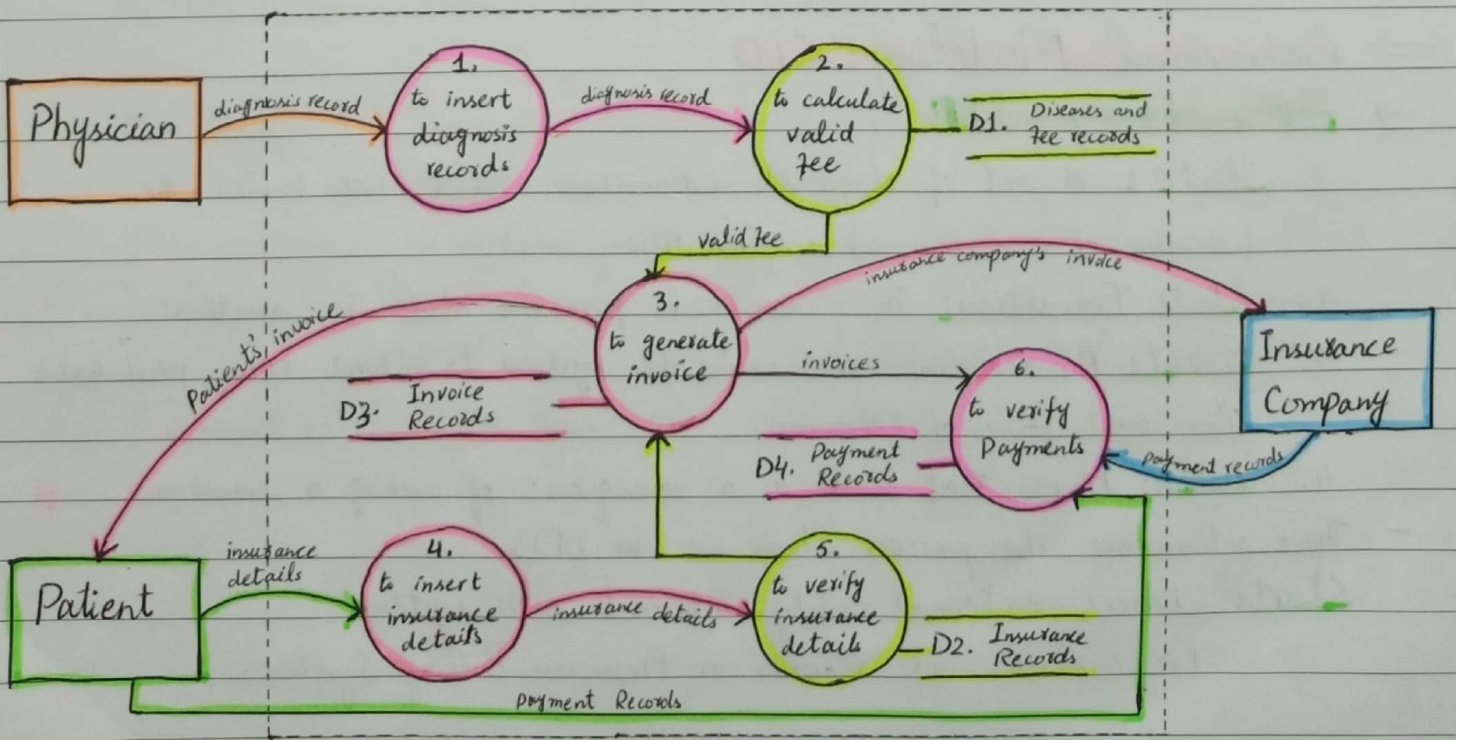
- It shows the direction of flow data.
- The direction of data is represented by an arrow.

→ **Physician Billing System**

○ Context Level or Zero Level DFD



○ Level 1 DFD



Date: 18 Oct, 2023 Lecture 13

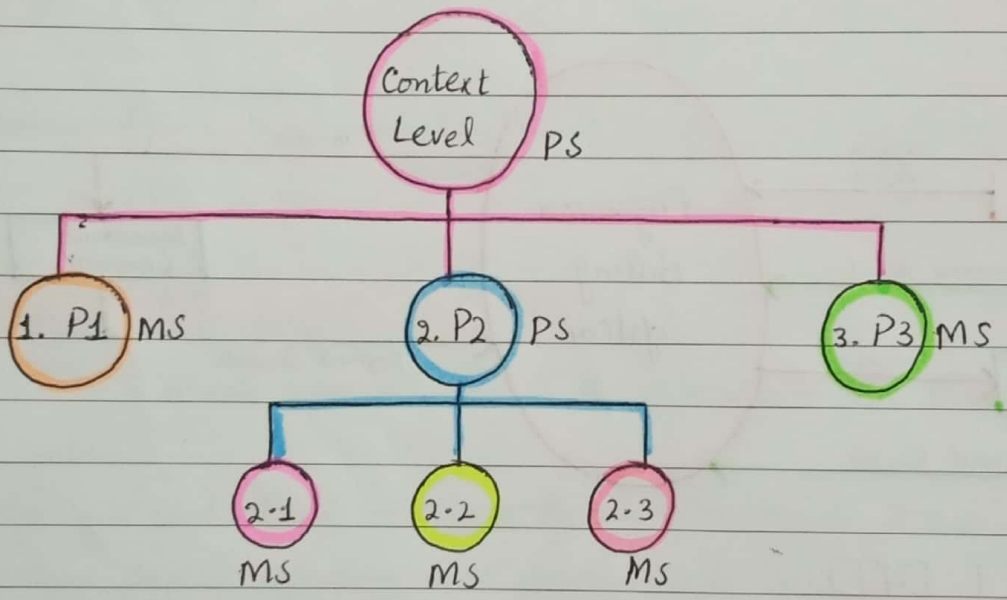
Data Structure and Algorithm

⇒ Documentation

- when we make ERD or DFD, we also have to maintain document of ERD or DFD for better understanding.

→ Documentation of Functional Modeling

- There are two types of documentation in DFD.
  - i. Process Specification (have further classification)
  - ii. Mini Specification. (no further classification) (leaves)



⇒ Behavioral Modeling: STD

→ Elements of STD

1. State: A set of observable circumstances that characterizes the behaviour of a system at a given time.
2. State Transition: The movement from one state to another
3. Event: An occurance that cause the system to exhibit some predictable (from one) form of behaviour.
4. Action: Process that occurs as a consequence of making a transition.

- These actions are the process which are in DFD.

Start: Where external entity interact with system.
For example all records in Physician Billing System.

Lecture 14 Date: 23 Oct, 2023

Software Engineering

⇒ Documentation of STD

- Control Specification is the documentation of STD.

⇒ Control Specification

State Name : Inserting Insurance Details
Source State : Start
Event: Insurance Details reached
Action: 'To insert Insurance Details'
Destination State: Generating Invoices

⇒ Data Dictionary

- Data dictionary is the documentation of ERD.
- Data dictionary is used to maintain entity and data members / variables / characteristics in the structure.

Name: Formal name of a system.
e.g Invoice

Aliases: Synonyms or other names.
i.e, reciept

Where Used: Which entity or process is consuming this
eg.

How Used: Used as input or output or bool data or as a decision maker

Description: Explaining entity. Six symbols are used
= (is composed of), + (AND), [|] (either or), { }ⁿ (n multiplication), (opt.) (optional data), "...text..." (comments.)

Format: Data Type

Date: _____

- After ERD, DFD and STD, when the analysis is complete, then we maintained a documentation which is known as Software Requirements Specification.
- And it contains two parts, known as functional requirements and non-functional requirements.

Lecture 15

Date:

Software Engineering

Before Mids.

Revision.

Date: 6 Nov, 2023 Lecture 16

Software Engineering

⇒ Design Concepts and Principles

Data Design (Data Structure & Algorithm) and (Database System)
Component-level Design
Architectural Design
User Interface Design

⇒ Fundamental Concepts

Abstraction: (Generic View) (Eye bird view)

- It is a high level of view anything.
- As the detail increases, abstraction decrease and refinement increase.

Refinement:

- Elaboration of detail for all abstraction.

Types of Abstraction:

1. Data Abstraction
2. Procedure Abstraction
3. Control Abstraction / Behavioral Abstraction

Divide and Conquer Rule

"Right man at right time for right task."

Modularity:

- degree/level of division for refinement
- What is "right" number of modules for a specific software design.

Types of Modules:

- There are four types of modules.
  - i- Input Module
  - ii- Output Module
  - iii- Processing Module
  - iv- Decision maker.
- Information sharing is not dependency.
- Modules must be independent in working.

→ Which thing can be affected by increasing no. of modules?

When number of module increases, cost of software decreases and

Date:

module integration cost increases.

Architecture:

- Architecture is the placement of the modules according to requirement.
- We have to arrange modules in shape of hierarchy.
- Also known as "Control Hierarchy" and "Program Structure".

- **Depth:** Vertical length / levels of modules
- **Width:** Horizontal length of modules
- **Fan in:** A position where mutiple super-ordinate control one subordinate. This is abnormal position, we should avoid it.
- **Fan Out:** A position in Control Hierarchy in which a module is controlling multiple modules is called Fan out position. And this is the normal position. In this position, there is one super-ordinate and multi-subordinate.
- **Connectivity:** Connectivity is the number of direct connections between executive module and the other modules
- **Visibility:** The executive modul can have visibility of all connected modules and control other modules.
  We can control visibility through control specifiers

Date: 8 Nov, 2023

Lecture 17

Software Engineering

⇒ Design Concepts

→ Functional Independence

- When we divide our software into modules then the association or connectivity between the modules should be such that they can work independently.
- The more modules/function work independently, the more increase the performance of software.

Types of Functional Independence:

1. **Cohesion:**
   - The degree to which a module (to another module) performs one and only one function.
   - The independence of one module to another module.
2. **Coupling**
   - The degree to which a module is connected to other modules in the system.
   - The dependence of one module to other modules.

∵ We need to keep modules more cohessive and loosely coupled as much as possible.

∴ Cohesion and coupling are resiprocal to each other.

"While developing the design, we try to maintain low coupling and high cohesion."

→ Association:

- Association is a relationship between two entities, where one entity is connected to another but they remain independent. For example: Library and Books, Person & Mobile

→ Dependency:

- Dependency implies that one class relies on other, suggesting a stronger connection where changes in one may affect the other. For example: Processer & Software, Lamp & Light Bulb.

Lecture 18

Date: 13 Nov, 2023

Software Engineering

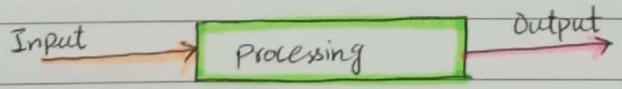
⇒ Architecture Design

- Architecture is the placement/arrangement of modules according to requirement.
- Take an example of map of a house. In map, we just design where will be the bedroom, washroom, kitchen, drawing room etc according to requirement. Similarly if we divide software into modules, such as which part will deal input (input controller), which part will deal with output (output controller) and which part will deal with processing (processing controller), then it will be the architecture design of software.
  And how these parts/modules will work, we don't have to concern in architecture in software like we don't concern about the inner design of drawing room, and bed room in map.

→ Types of Flows

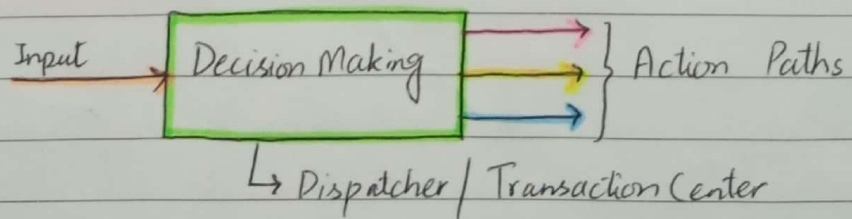
1. Transformation Flow

- pure processing.



2. Transaction Flow

- Pure decision making



- If multiple paths can occure and can be selected from a processor at a same time, then the flow is pure transformation flow.

Date:

→ 3 Situation can Occur

1. Pure transformation flow
2. Pure transaction flow
3. Both

⇒ String Conversion System

Lecture 19

Date: 15 Nov, 2023

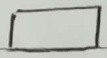
Software Engineering

⇒ Component Level Design

- Define logic in every module. And every module will be treated as individually.
- Component-level design is considered the "closest activity" to the coding.

Tools to Make Component-Level Design Model

1. Graphical

- Flow Chart
  - Every problem in the word can be resolved using three symbols/statement
    - i- Sequence Statement 
    - ii- Conditional Statement 
    - iii- Repititional Statement (no symbol) [Adjust/manage by arrow head]
- These three statements known as "Programming Conistructs".
- Flow chart is mostly used to show the Process.

2. Pseudocode and Programming Language and Algorithm

- Pseudocode is also known as PDL (Programming Design Language.
- Algorithms only contains logic, and are language independent.
- Pseudocode is the mixture of logic and programming language.
- In algorithms, it is not neccessary to use programming language. But in pseudo code, there is presence of programming language.

3. Decision Table

If (C1 && C2)

no. of combination = 2^n

of conditions.

- Desision table helps to overcome the problem of finding combination of conditions.

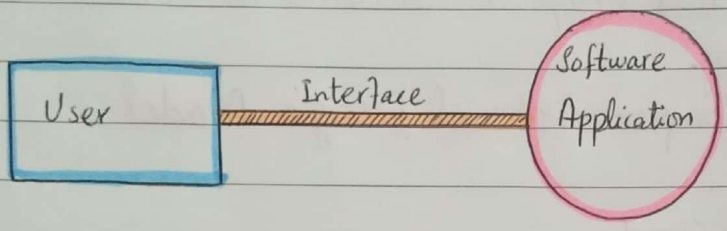
Date: 20 Nov, 2023

Lecture 20

Software Engineering

⇒ User Interface Design

- Interface is the communication or usage medium for user.
- It plays a role of bridge between software and user.



→ User Interface Problems

1. Control Problem
2. Consistency (when an app update, same or different user interface)
3. Memory Load (Memory load should be lowest on user)
4. No Guidance (Not a user-friendly interface)
5. Poor Response

- There should be consistency in user interface, because humans are reductant to accept change abruptly.

→ Golden Rules: (Interview Question)

1. Place the user in control.
2. Reduce the load of memory on user.
3. Make the app interface consistent.

→ User Interface Design Process

1. User, task and environmental analysis
   - All human tasks are defined and classified.
   - Objects and actions are identified for each task.
   - Redefine tasks iteratively throughout the process.
2. Interface Design

   HCI → Human Computer Interaction.

   1. Site map
   2. Story board
   3. D

Date: 22 Nov, 2023 Lecture 21

Software Engineering

⇒ Object Oriented Analysis and Design

- Why we need object oriented approach when we have structured approach?
  - There was no simultaneously control of data and functionality in structured approach.

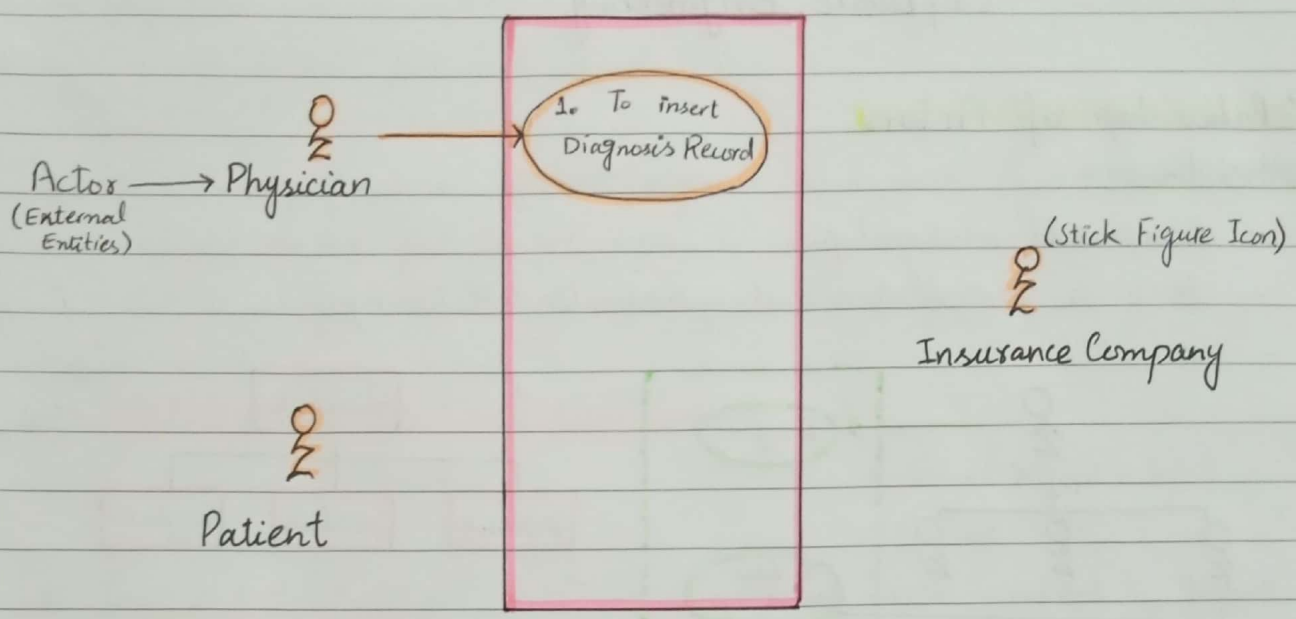
1. Use Case Modelling
   - High Level Use Case Diagram
   - Analysis Level Use Case Diagram
   - Use Case Description (Documentation)
2. Domain Model
3. System Sequence Diagram
4. Sequence Diagram
5. Design Class Diagram

⇒ Use Case Modelling

The goals of external entities of Physician Billing System
- Physician (provide diagnosis records)
- Patient (submit insurance form and payment details)
- Insurance Company (provide payment details)

- we just need to show high level goals. We are not concerns how to achieve those goals.

Date:



→ **EBP**: (physical Level) (Runtime)

Use Case (class) ↓ EBP (object) ← { Physician A, Patient X, Diagnosis Flu

∵ EBP is also called "Concrete Use Case" or "Real-time Use case"

(Elementry Business Process)

→ **Actors**

- In Use Case modelling, external entities are called actors.

Student, Teacher, Degree Coordinater, Program Coordinater, Parents (CMS)
↓ primary, ↓ primary, ↓ secondary, ↓ secondary, ↓ offset

1. **Primary Actors**:
   - Actors with main goals are called primary actors.
2. **Secondary Actors**:
   - Actors who assist to achieve main goal are called secondary actors.
3. **Off Stage Actors**:
   - Actors who have no primary goals, and neither assist to achieve the goal, but have the interest in the information that hase been generated.

Use Case Modelling must have

1. Goal ⬭
2. Actor
3. Scenario (EBP)
4. System ▭

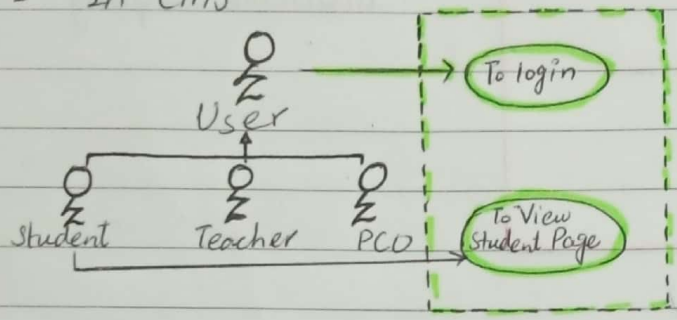
Date: 27 Nov, 2023 Lecture 22

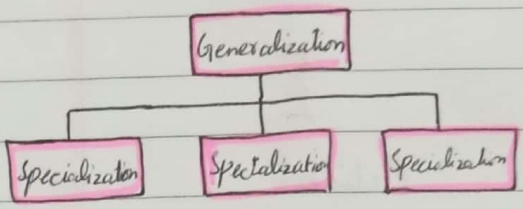
Software Engineering

⇒ Relationship of Actors

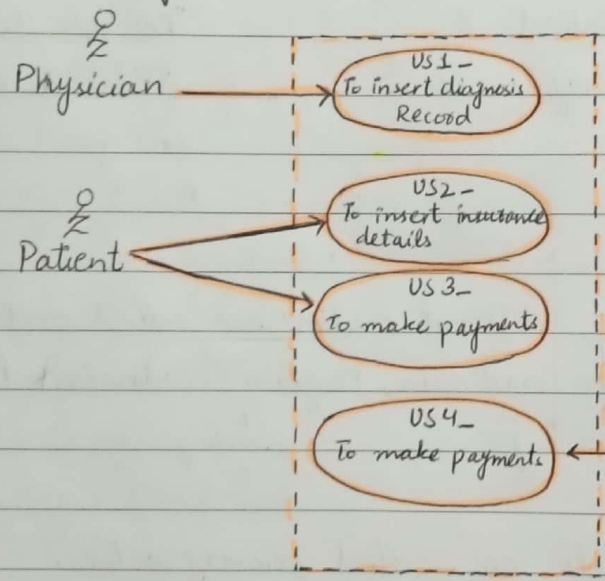
→ Inheritance:

- Parent-child relationship.
- It is is generalization and specialization relationship.
- In CMS

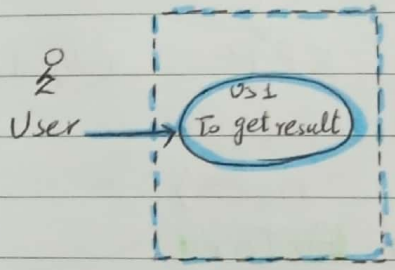




In Physician Billing System



In String Conversion System



General Talk

→ **Vision:** (It is all about future).

- To think and see where will you be at after 10 years or some years.
- The vision of organization remain same but for person it can change.

→ **Mission:**

- What are you currently doing for the achievements of what you want to become or to achieve vision.

→ **Goals:**

- To achieve mission.
- Every goal have sub-goals.

Date:

→ Objectives:

- To achieve, goals and subgoals.

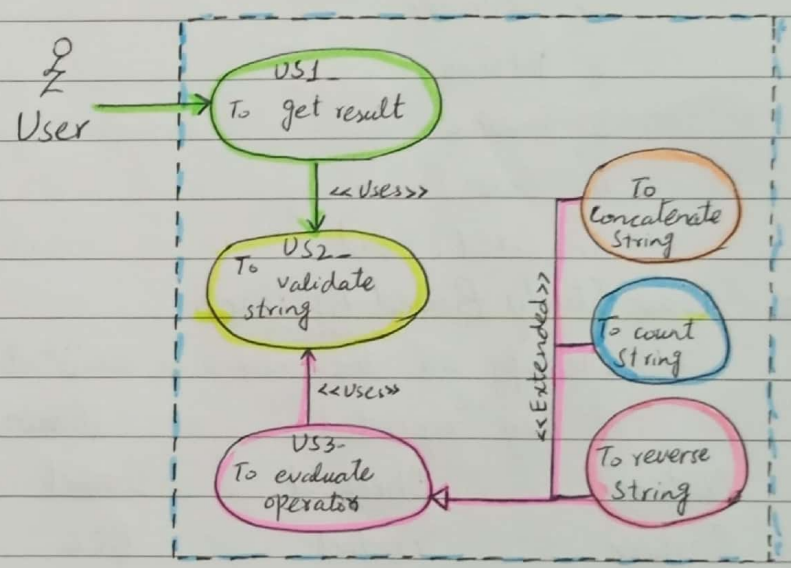
∴ To see myself as a software engineer is a vision. To become a software engineer is my goal. To get degree to become a software engineer is mission. And to study 8 semester to achieve goal are objectives of goals.

⇒ Relationship Between UseCases:

{1. Internal Relationship (Mandatory 2. External Relationship (Optional}

1. Included Relationship
   - Mandatory

2. Extended Relationship
   - Optional



⇒ UseCase Description:

1. Brief (one paragraph) 2. Casual (multi-paragraph) 3. Fully Dressed

- UseCase Name
- Primary Actors
- Stakeholders and interests
- Preconditions
- Success Guaranties (Post Condition)
- Main Success Scenario
- Extensions (Alternative Flow)
- Special Requirements
- Technology and data version list
- Open Issues

Date: 29 Nov, 2022

Lecture 23

Software Engineering

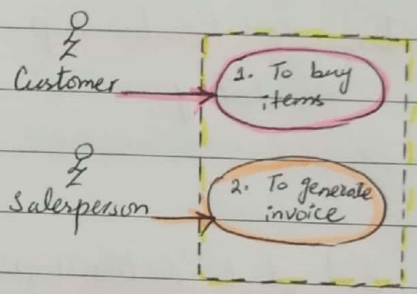
⇒ Domain Model

- Domain model is the first step in object oriented approach in which we make and identify classes and associate classes with each other according to requirements by using UseCase modelling.
- The goal of domain model is to identify classes and relate them.

⇒ Approaches to Identify Classes

1. UseCase Based Approach

- In point of sale system, classes will be following
  1. Customer
  2. Items
  3. Invoice
  4. Sales Person
  5. Credit Card



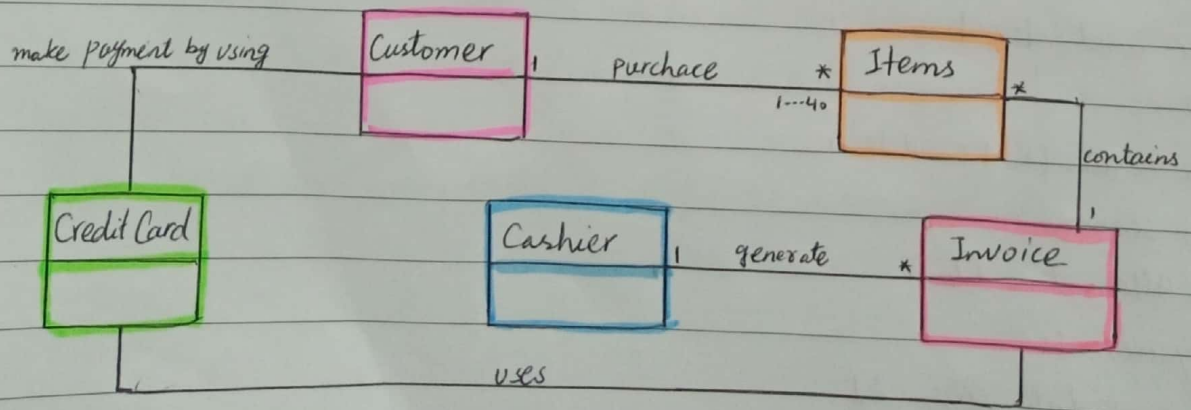
2. Case Study Based Approach

- Identify all the nouns in case study and underline them even if they are repeated. (Brain storming)

Customer	Checkout	Basket	Goods
Cashier	Product	Price	Items
System	Price	Transaction	Customer
Reciept	Total	Customer	Payment
Credit Card	Basket	Card	

- Remove duplicates now.

Customer Items Credit Card Reciept Cashier



Date:

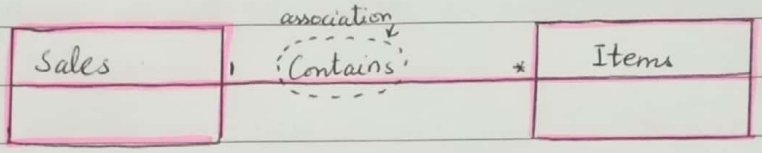
- Domain model classes are conceptual classes. And Conceptual classes are integral part of final solution

→ Domain Model Terms

- Following are some terms that are commonly used in domain model.

- Association:

- The sementic relationship between two or more classes that involves interaction/ connection among their instances.
- Association name should be a verb phrase.



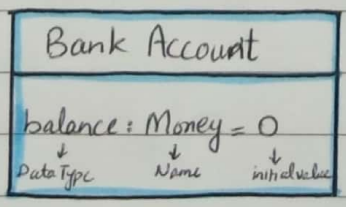
- Multiplicity:

- Multiplicity is how many instances of a class A can be associated with the instance of class B in a particular moment.

*	represent many instances
1	exactly one instance
1...*	one or more instances
1...40	instances between 1 to 40
3,8	exactly 3 or 8 instances

- Attributes

- An attribute is a logical value of an object
- An attribute is defined by
  - Name (Noun)
  - DataType (character, integer (numeric), alphanumeric, boolean)
  - Initial value.



- DataType and initial value are optional.

Date: 04 Dec, 2023 Lecture 24

Software Engineering

General Talk

When you want advice / suggestion / consel from a person.

- That person should be expert in that field.
- That person should be experienced in that field.
- That person should be characterfull.

Lecture 25

Date: 6 Dec, 2023

Software Engineering

⇒ **Object Oriented Design:**

⇒ **Object Design**

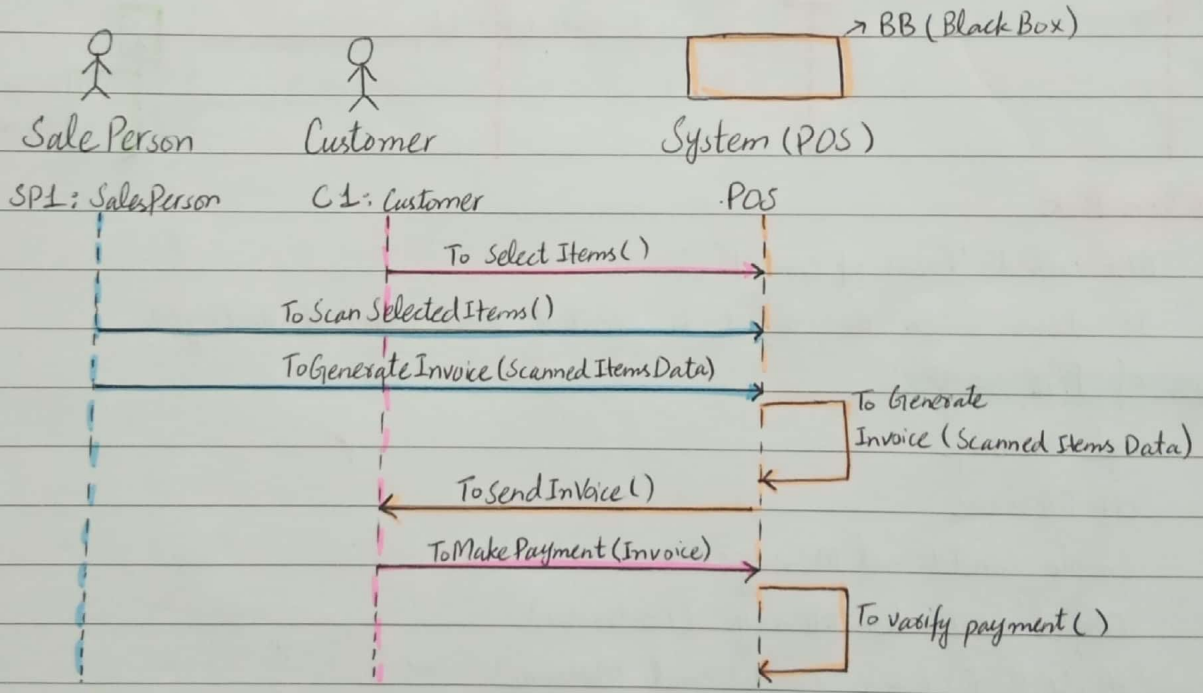
:→ **Interaction Diagram:**

1. System Sequence Diagram (SSD) (same as context level DFD)
2. Sequence Diagram
3. Collaboration Diagram

In above two diagrams, we will take entities as object.

Example: Point of Sale person.

- Domain mode will be used here as an input.
- We will take system here as "Black Box."

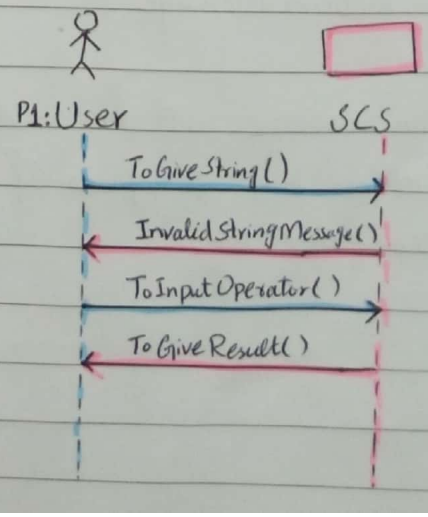


System Sequence Diagram

- The dotted lines show the lifetime of object.

String Conversion System:

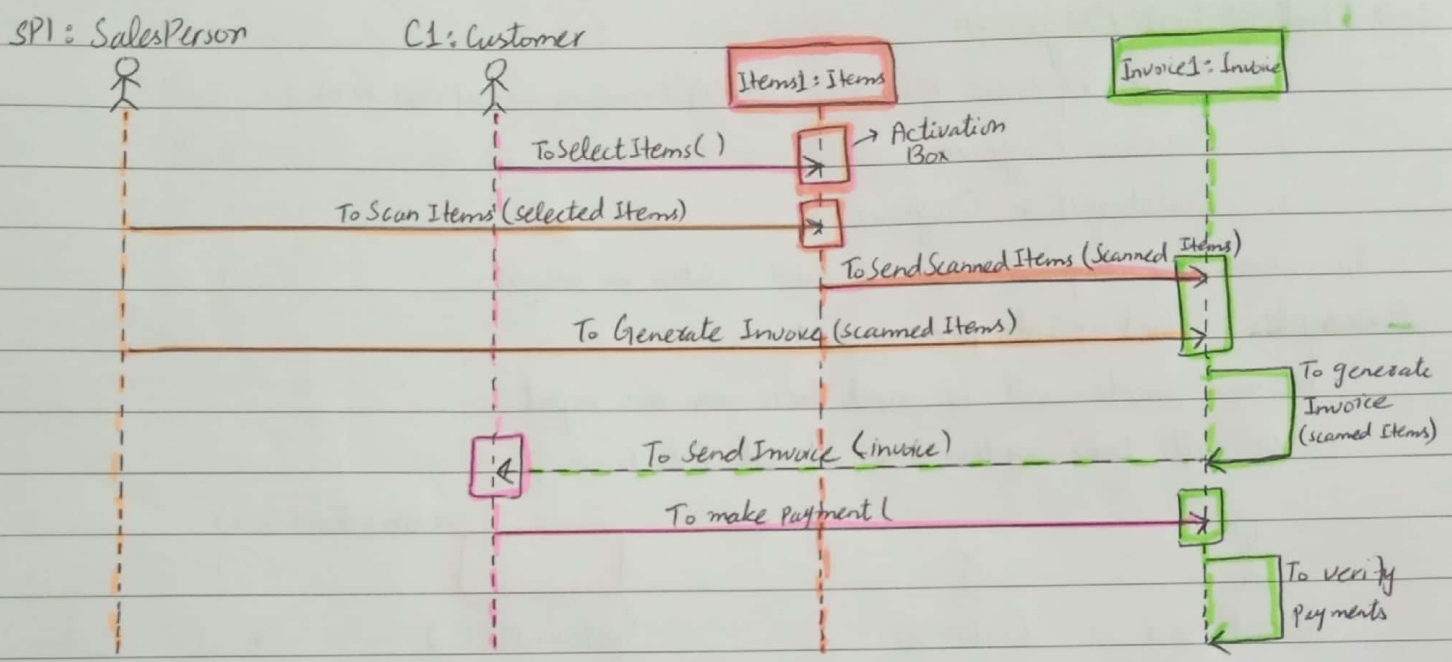
- There's only one class for whole system.



Date:

2. Sequence Diagram

- In this diagram, we open the Black Box System.
- In this diagram, objects are shown, not the entities/classes.



Activation Box:

- Also called 'Focus of control'
- It shows when the object is created and when it destroyed.

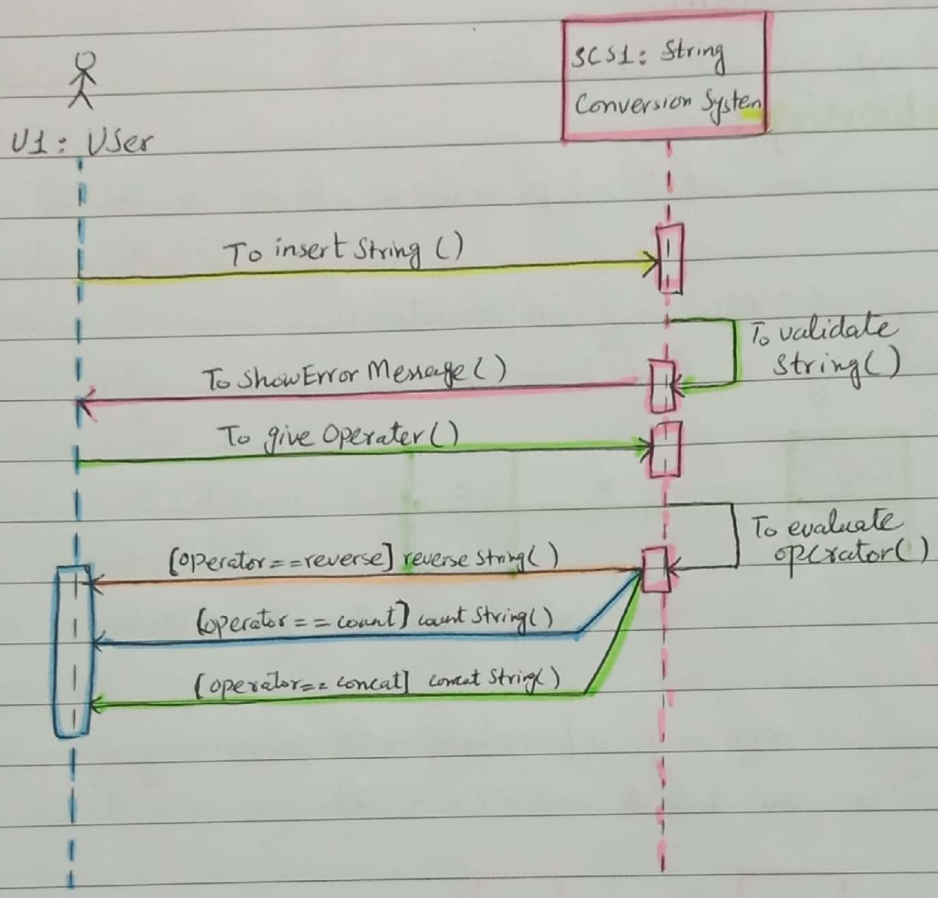
→ Types of Messages

- Simple Message
- Self Message
- Simple Conditional Message
- Object Destruction Message (Destructor)
- Mutually Exclusive Conditional Message
- Return Message

- Return message will be in dotted lines.
- Activation box or focus of control is represented by vertical rectangular box.

Date: _____

String Conversion System



Date: 11 Dec, 2023 Lecture 26

Software Engineering

⇒ Design Class Diagram

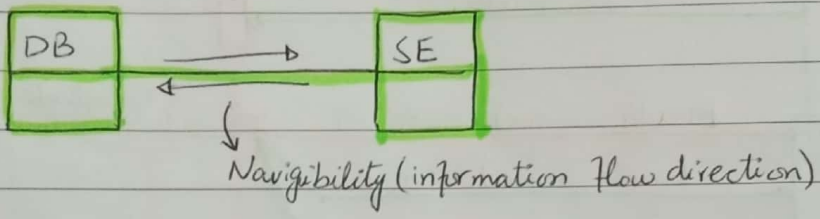
- Relationship between classes.

→ Class Relationship:

Following are some relationships between classes in DCD.

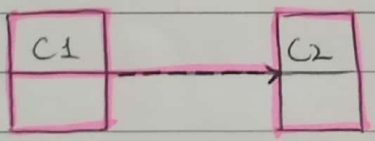
1. Association

- Conveying or recieving information
- Represent by a line (——).



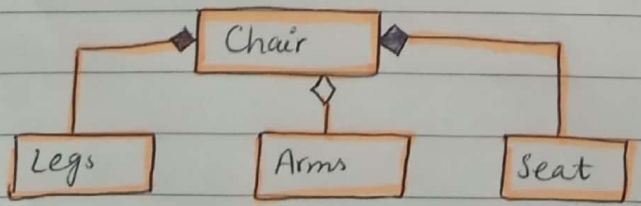
2. Dependency

- Conveying or recieving information with processing.
- Represent by ~~an~~ dotted arrow (----→)



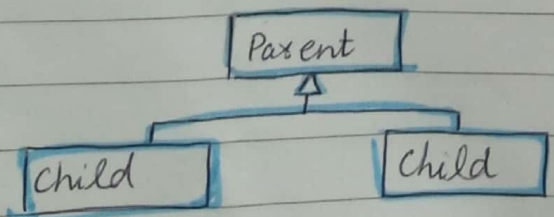
3. Containment

- whole part relationship
- Composition (strong) (represent by filled diamond ◆)
- Aggregation (weak) (represent by empty diamond ◇)



By value → composition
By refrence → Aggregation

4. Inheritance



Lecture 27

Date: 13 Dec, 2023

Software Engineering

⇒ Design Pattern

Pattern: Repition of instance after a specific time periods.

Definition:

- The solution of re-occuring problem is known as "design pattern."

→ Capacity, Ability, Competency
→ Authority (Connectivity, Visibility) (Scope Specifier) (Power)
→ Responsibility

} Role of Class

- These three are includes in every class.
- Now we will discuss these three in two angles

General Life

- Your authority is to use the appropriate usage of resources by using your capacities/abilities to fullfill your responsibility.
- In general life, decision making is on run-time.
- It can be varied time-to-time.

S/W Development

- It is static.
- Every class has capacity, authority and responsibility. Authority is the visibility of data and methods in the class.
- Authority is used by objects to fullfill its responsibility.

- In software development, when you are declaring the data members and methods, you're actually declaring the capacity and authority of class.
- When Capacity, Authority and Responsibly joins it make "Role."

GRASP

- General Responsibility Assignment Software Patterns
- Basic patterns of software engineering.