LECTURE 1

INTRODUCTION TO SOFTWARE ENGINEERING

Software Engineering

The process of ***solving customer's problems*** by the ***systematic development and evolution*** of ***large, high-quality software systems*** within ***cost, time and other constraints.***

Fritz Bauer

German Jewish judge and prosecutor, instrumental in the post-war capture of Adolf Eichmann and played an essential role in beginning the Frankfurt Auschwitz trials.

Solving customers' problems

- The ***goal*** of software engineering is to ***solve customer problems***
- Sometimes the solution is to ***buy, not build***

Systematic development and evolution

- An engineering process involves applying well understood techniques in an organized and disciplined way.
- Most development work is ***evolution***.

  def. The adaptability of the software, in terms of hardware, OS, etc.

Large, high quality software systems

- Software engineering techniques are needed because large systems cannot be completely understood by one person
- Teamwork and coordination are required
- Key challenge: Dividing up the work and ensuring that the parts of the system work properly together
- The end-product must be of sufficient quality

Cost, time and other constraints

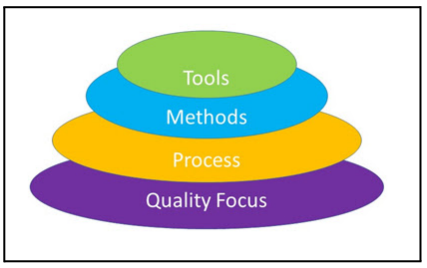
- Finite resources
- The benefit must outweigh the cost

Software Engineering- A Layered View

The concept to understand software engineering is through layered view.

This view consists of ***four layers:***

- ***Tools***
- ***Methods***
- ***Process***
- ***Quality Focus***



Quality Focus

- The very ***foundation*** of the layer
- A culture where commitment to continuous improvement on the software development process is fostered.
- Enables the development of more effective approaches to software engineering.

Process

- ***Integrates the other layers together.***
- Defines a framework that consists of key process areas that define and enable rational and timely delivery of the computer software
- ***Key process areas***

  def. The basis for the software project management
- Establishes:
  - what ***technical methods*** are applied
  - what ***tools*** are used
  - what ***work products*** need to be produced
  - what ***milestones*** are defined
- Include assurance that quality is maintained, and that change is properly controlled and managed

Method

- Define a ***systematic and orderly procedures of building software***
- Provides an ***overall framework within each activities*** of the software engineer are performed
- These ***activities*** include a wide array of tasks such as:
  - ***Requirements analysis***
  - ***Design program construction***
  - ***Testing and maintenance***
- ***Methodology***

  def. The science of ***systematic thinking*** using the methods or procedures used in a particular discipline
- Several ***software engineering methodologies*** that are used today:
  - ***Structured Methodologies***
    - Information Engineering
    - SDLC
    - Rapid Application Development Methodology
    - Joint Application Development Methodology
    - CASE*Method
  - ***Object-oriented Methodologies***
    - Booch Method
    - Coad and Yourdon Method
    - Jacobson Method
    - Rambaugh Method
    - Wirfs-Brock Method

Tools

- Provide ***support*** to the process and methods
- May be ***automated*** or ***semi automated***
- Most are used to ***develop models***
- ***Models***

  def. ***Patterns*** of something to made or simplification of things

  - ***Two Models*** that are generally developed by a SE:
    - ***System Model***
      - Inexpensive representation of a complex system that one needs to study
    - ***Software Model***
      - Blueprint of the software that needs to be built

- ***Structured Approach Modeling Tools***
    - Entity-relationship Diagrams
    - Data Flow Diagrams
    - Structured English or Pseudocodes
    - Flow Charts

- ***Object-oriented Approach Modeling Tools***
    - Unified Modeling Language (UML)

Quality with the Development Effort

- ***Quality*** in all software engineering activities ***reduces costs*** and ***improves time-to-market*** by minimizing rework.
- Software engineer must explicitly ***define what software quality is***, ***have a set of activities*** ensuring that every ***software engineering work product exhibits high quality***, do ***quality control and assurance activities***, and ***use metrics*** to develop strategies for improving the software product and process

Quality

- The ***totality of characteristics of an entity*** that bear on its ability to satisfy stated and implied needs
- About meeting user requirements, providing what is wanted, when it is wanted, every time
- Building software that meets user needs and requirements that is delivered on time and within budget

Three Perspectives on Quality

- ***Quality of the Product***
    - Software is built based on end-user's external requirements combined with the developer's internal requirements
- ***Quality of the Process***
    - When a task or activity within the process fails, the quality of the product fails
    - Supports the culture of quality that influences the development of quality software
- ***Quality in the Context of the Business Environment***
    - Software must add value to the business

Addressing Quality Issues

- **Quality Standards**
    - Sets of principles, procedures, methodologies, and guidelines to bring about quality in business processes
        - CMMI (Capability Maturity Model Integration)
        - ISO 9000:2000 for Software
        - Software Process Improvement and capability Determination (SPICE)

- **Understanding Systematic Biases in Human Nature**
    - People tend to be risk averse when there is a potential loss.
    - People are unduly optimistic in their plans and forecasts.
    - People prefer to use intuitive judgment rather than quantitative models.

- **Requires commitment from the whole organization and everybody involved in the software development effort.**
    - A change in mindset is required; A bug should be fixed immediately upon detection.

- **User requirements should be managed because it changes overtime.**

Software Quality Assurance and techniques

- A ***subset of software engineering*** that ensures that ***all deliverables and work products are met***, and they comply with user requirements and standards.
- Considered as ***one of the most important activities*** that is applied throughout the software development process.
- ***Goal of Software Quality Assurance:*** To detect defects before the software is delivered as a final product to the end-users.

Software Quality

- A software that is ***fit to use***, ie, it is ***working properly.***
- Meets users' ***requirements*** (must be measurable in order to gauge compliance).
- Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

Three Important Points

- **Software Requirements** are the ***foundation from which quality is measured***. It is necessary to explicitly specify and prioritize them.
- **Standards** define a ***set of development criteria*** that guide the manner by which software is engineered.
- ***Implicit requirements*** must be ***identified and documented***; they influence the way software is developed such as maintainability.

Characteristics of a Good Quality Software

- **Usability**
    - A characteristic of the software that exhibits ***ease with which the user communicates with the system***.
    - Users can learn it fast and get their job done easily.

- **Portability**
    - Capability of the software to ***execute in different platforms and architecture***.

- **Reusability**
    - Ability of the software to ***transfer from one system to another.***
    - Its ***parts can be used in other projects***, so reprogramming is not needed

- **Maintainability**
    - Ability of the software to ***evolve and adapt to changes*** over time.
    - Characterized by the ***ease of upgrading and maintaining.***
    - Can be ***easily changed***

- **Dependability**
    - A characteristic of the software to be ***reliable, secure and safe***

- **Efficiency**
  - Capability of the software to ***use resource efficiently***
  - ***Doesn't waste resources*** such as CPU time and memory

- **Reliability**
  - It ***does what it is required to do*** without failing

LABORATORY 1

Java

- Widely used OOP language and software platform that runs on billions of devices
- Rules and syntax of Java are based on the C and C++ languages.
- One major advantage of developing software with Java is its ***portability.***
- Invented in ***1991*** by ***James Gosling*** of ***Sun Microsystems*** (late acquired by Oracle)
- Primary goal: ***"write once, run anywhere"***

Java vs Javascript

- Java is different from Javascript
- Javascript needs to be compiled, Java code does need to be compiled.
- Javascript only runs on web browsers , java can be run everywhere.

How Java works

The ***Java software platform*** consists of the ***JVM***, the ***Java API,*** and a ***complete development environment.***

Java Development Kit (JDK)

- To create an application using Java
- Available for Windows, maccOS, and Linux.

Java Virtual Machine (JVM)

- Part of **Java Runtime Environment (JRE)**
- Parses and runs the Java bytecode

Java API

- consists of an ***extensive set of libraries*** including basic objects, networking and security functions; Extensible Markup Language (XML) generation; and web services.

A ***container object*** that holds a fixed number of values of a single type, starts with 0, an arrangement of numbers, pictures or objects formatted into rows and columns
according to their type

HTML

- The standard markup language ***for documents designed to be displayed in a web browser.***
- It ***defines the meaning and structure*** of web content.
- It is often assisted by technologies such as ***Cascading Style Sheets*** and scripting languages such as ***JavaScript***.

Static vs Dynamic Website Design

Dynamic Website

- ***Contains information that changes*** depending on the time of day, the viewer, time zones, the viewer's native language, and other factors.
- Can provide ***user interaction***, ***display different content***, and can be more intriguing. The content can change for each viewer.
- Can be quite difficult, as it has various components that have to be considered.

Static Website

- ***Contains information that doesn't change***. It stays the same for each viewer that visits the website.
- Generally comes with a fixed number of pages that have a ***specific layout***. When a page runs on a browser, the content is static and ***doesn't change in the response to the user.***
- Not as complicated as a dynamic website design

LESSON 2

Activities establish a skeleton architecture for software engineering work.

- **Communication**
- **Planning**
- **Modeling**
- **Construction**
- **Deployment**

The Software Process

Process is a ***collection of activities, actions, and tasks*** that are ***performed when some work product is to be created.***

Activity strives to ***achieve a broad objective*** and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.

Action encompasses a ***set of tasks that produce a major work product***

Task focuses on a ***small, but well-defined objective*** that ***produces a tangible outcome***.

Software Design Paradigm

This paradigm is a part of Software Development and includes:

- ***Design***
- ***Maintenance***
- ***Programming***

Programming Paradigm

This paradigm is closely related to the programming aspect of software development. This includes –

- ***Coding***

- ***Testing***
- ***Integration***

Software Development Life Cycle

A well-defined, structured ***sequence of stages in software engineering*** to develop the intended software product.

SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently.

SDLC framework includes the ***following steps:***

Communication

- The user contacts the service provider and tries to negotiate the terms, submits the request to the service providing organization in writing.

Requirement Gathering

- The team holds discussions with various stakeholders from the problem domain and tries to bring out as much information as possible on their requirements.
- The requirements are contemplated and segregated into user requirements, system requirements and functional requirements.

Feasibility Study

- The team comes up with a rough plan of the software process.
- There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis

- The **developers decide a roadmap of their plan** and try to bring up the best software model suitable for the project.

Software Design

- Bring down whole knowledge of requirements and analysis on the desk and design the software product.
- The inputs from users and information gathered in the requirement gathering phase are the inputs of this step.
- The output of this step comes in the form of ***two designs***;
  - ***Logical Design***
  - ***Physical Design***
- Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams, and in some cases pseudocodes.

Coding (aka Programming Phase)

- The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

- 50% of the whole software development process should be tested.
- Errors may ruin the software from critical level to its own removal.

- Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing

Integration

- Software may need to be integrated with the libraries, databases, and other program(s).
- This stage is involved in the integration of software with outer world entities.

Implementation

- Installing the software on user machines.
- Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

- Confirms the software operation in terms of more efficiency and less errors.
- If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational.
- The software is maintained timely by updating the code according to the changes taking place in the user end environment or technology.
- This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition

- Plans are developed for discarding system information, hardware, and software and making the transition to a new system.
- The information, hardware, and software may be moved to another system, archived, discarded, or destroyed.

Classical Waterfall Model

- The ***simplest model*** of software development paradigm.
- All the phases of SDLC will function one after another in a ***linear manner.***
- ***Major drawback:*** We move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages.

Iterative Model

- Leads the software development process in ***iterations.***
- Projects the process of development in a cyclic manner ***repeating every step after every cycle*** of the SDLC process.

Spiral Model

- ***Combination*** of both, ***iterative model and one of the SDLC models***.
- It can be seen as if you choose one SDLC model and combine it with a cyclic process (iterative model).

Prototyping Model

- Prototypes
  - Partially developed software that ***enables end-users and developers to examine aspects of the proposed system*** and decide if it is included in the final software product.
  - Built to ***aid in the understanding of end-user requirements.***

Rapid Application Development (RAD) Model

- A ***linear sequential*** software development process that emphasizes an extremely short development cycle.
- It is achieved through a ***modular-based construction approach***.
- It is best used for software projects where requirements are well- understood, project scope is properly constrained and big budgets with resources are available.

V-model (Testing phase)

- V-Model provides means of testing of software at each stage in ***reverse manner.***

Software Development Paradigm

- Also known as **Software Engineering;** all engineering concepts pertaining to developments software applied.
- Part of Software Development
- Consists of the following ***parts:***
    - **Requirement gathering**
    - **Software Design**
    - **Programming**

Extreme Programming

- uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of ***four framework activities:***
    - ***Planning***
    - ***Design***
    - ***Coding***
    - ***Testing.***

Software Aging Causes

- ***Two types of Software Aging:***
    - ***Lack of Movement***
        - Aging caused by the failure of the product's owners to modify it to meet changing needs.
    - ***Ignorant Surgery***
        - Aging caused as a result of changes that are made.

LABORATORY 2

Review nyo yung Lab 2 na binigay nya, yung output giving emeru. Di ko na para ilagay yon dito matuto kayong tumayo sa sarili nyong paa.

https://drive.google.com/file/d/1XUYXP6L9kxsjzCXOMHDbH2qax4u3FUbe/view

Ayan na yung link ha alam kong tamad kayo maghanap. Yung output nyan malamang nasa activity nyo.

LECTURE 3

SOFTWARE PROJECT MANAGEMENT

Project

A set of **activities to be performed within a certain time frame** using defined resources.

Can be characterized as:

- May have a **unique and distinct goal**
- Is **not routine activity** or day-to-day operations
- Comes with a **start and end time**

Project **ends when its goal is achieved** hence it is a temporary phase in the lifetime of an organization

Project needs **adequate resources** in terms of time, manpower, finance, material and knowledge-bank.

Project Management

Planning, monitoring and control of the people, process and event.

SOFTWARE PROJECT MANAGEMENT SCOPE

4P's in Software Project Planning

- **People**
- **Process**
- **Product**
- **Project**

Project Manager

- A person who **undertakes responsibility of executing the software project**
- **Thoroughly aware of all the phases of SDLC** that the software would go through
- **May never directly involved in producing the end product** but controls and manages the activities involved in production

Software Team

Democratic Decentralized

- No permanent leader
- Team Leader is also a coordinator
- Require team coordination
- Group decisions are followed

Controlled Centralized

- One component is designated as the controller and is responsible for managing the execution of other components.
- Two Classes:
    - Sequential
    - Parallel

Controlled Decentralized

- Has a **defined leader** that **coordinates specific tasks**
- Has a **secondary leader** that have the **responsibility of subtasks**
- **Vertical communication** along the control hierarchy

Software Project - People

Motivation

- Achievement
- Recognition
- Advancement / Promotion
- Salary
- Possibility for growth

Factors

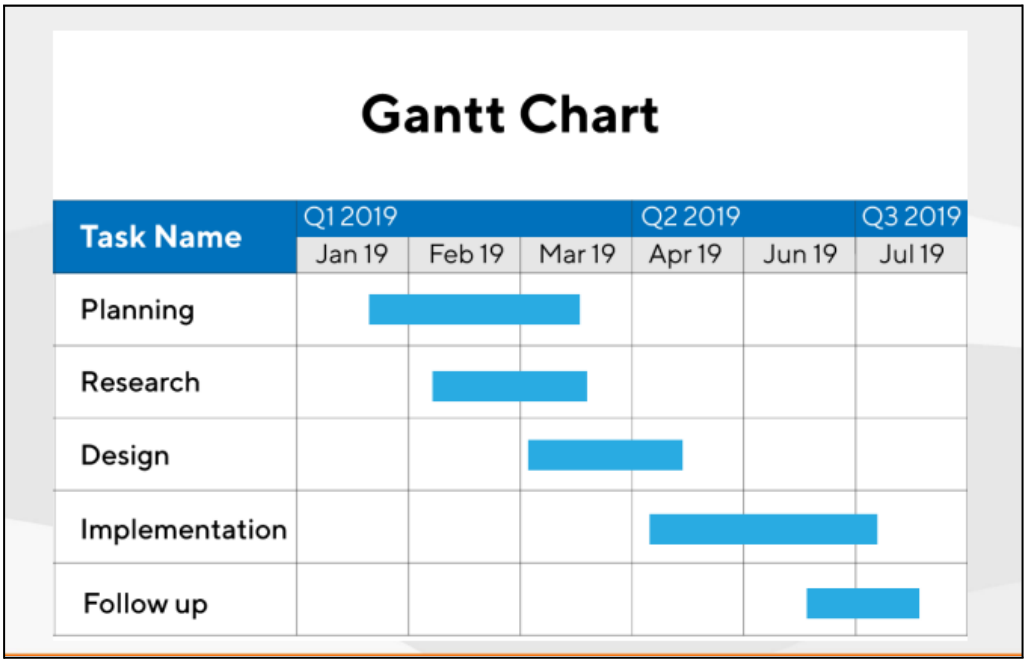
- Interpersonal relationships
- Subordinate
- Superior
- Peer
- Status
- Technical supervision

Project Management Tools

Project Management Tools available which aid for effective project management

- Gantt Chart
- Resource Histogram
- PERT Chart
- Critical Path Analysis

Gantt Chart



Resource Histogram

Contains a **bar or chart** representing the **number of resources required over time** for a project event. It is an effective tool for staff planning and coordination.

PERT Chart (Program Evaluation and Review Technique)

Depicts project as a network diagram. It is capable of graphically **representing main events** of a project in both parallel and consecutive ways. Events, which occur one after another, **show dependency of the later event over the previous one.**

Critical Path Analysis

Useful in recognizing interdependent tasks in the project. It also helps to **find out the shortest path or critical path** to complete the project successfully.

Object Modeling using UML

Unified Modeling Language (UML)

An **industry-standard graphical language**, specifying, visualizing, constructing and documenting the artifacts of software system.

Combines technique from:

- Object modeling
- Data modeling
- Business modeling
- Component Modeling

Modeling

A diagram with graphical representation of system model

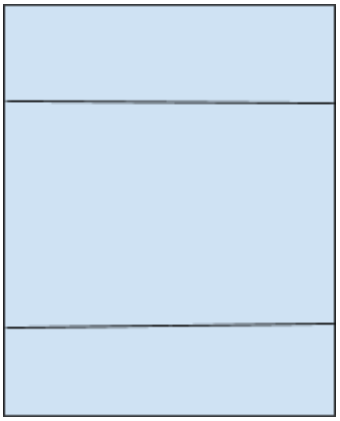
2 Different Views of System Model (represented by UML diagram)

- **Static** (Structural Model)
- **Dynamic** (Behavioral Model)

Class Diagrams

A UML Class Diagram:

- Describe classes and their relationship
- Class is denoted as a **rectangular shape box** usually divided into three parts.
- **Classes** define the blueprint for an object
- **Object** are custom variables which contain state and behavior
- **State** is represented as properties and behavior as methods.



Class name (should always be in uppercase letter first)

Attributes (also called fields, variables)

Behaviors (Methods)

Visibility

Set the **accessibility** for the attributes and method

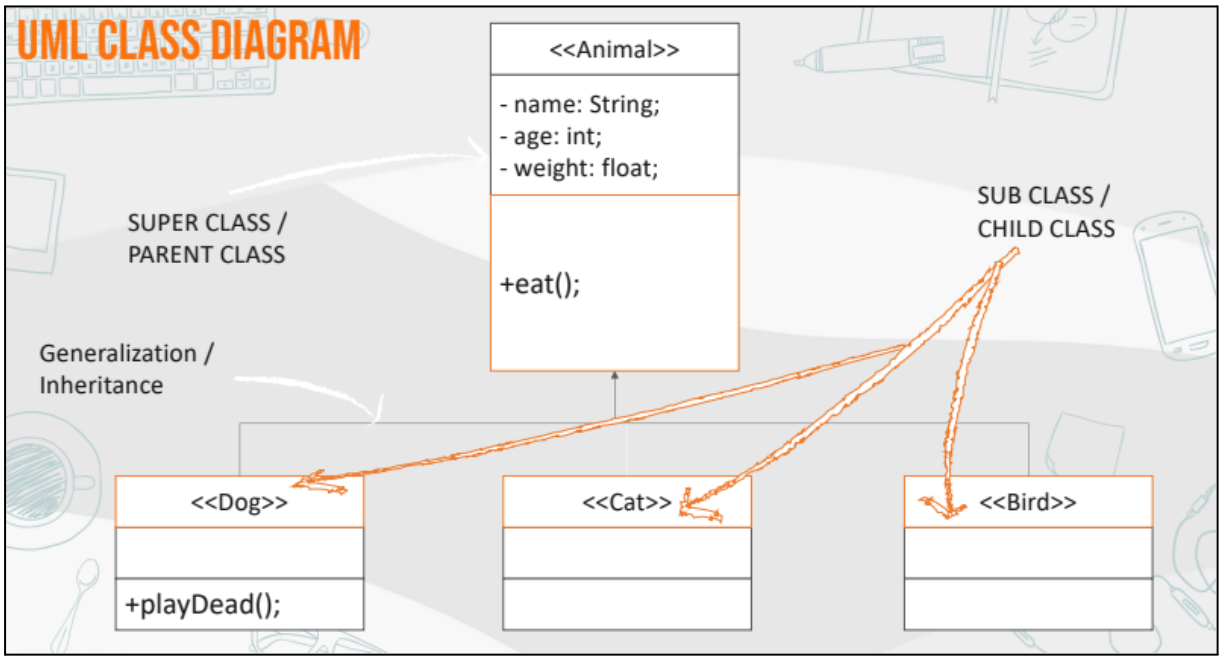
- \- Private
- **+** Public
- **#** Protected
- **~** Package/Default

UML Class Relationship

Generalization/Inheritance

Known as an **"is a"** relationship since the **child class is a type of the parent class**. Generalization is the ideal type of relationship that is used to showcase reusable elements in the class diagram. Literally, the child classes **"inherit"** the common functionality defined in the parent class.

Inheritance Animal Example



Association

An Association that exists between classes is **bi-directional**. Ideally, you may illustrate the flow of the association by utilizing a directed association.

Aggregation

When a **class is formed as a collection of other classes**, it is called an aggregation relationship between these classes. It is also called a **"has a"** relationship.

Composition

A variation of the aggregation relationship. Composition illustrates that a **strong life cycle is present between the classes.**

Multiplicity

The relationship shows a star sign (*) (one to many and many to many etc).

Review of OOP Concepts

Inheritance

Is a mechanism in which **one object acquires all the properties and behaviors of a parent object.**

Abstraction

Is a process of **hiding the implementation details** and showing only functionality to the user.

Polymorphism

A concept by which we **can perform a single action in different ways**. Derived in Greek words “poly” (many) and “morphs" (forms).

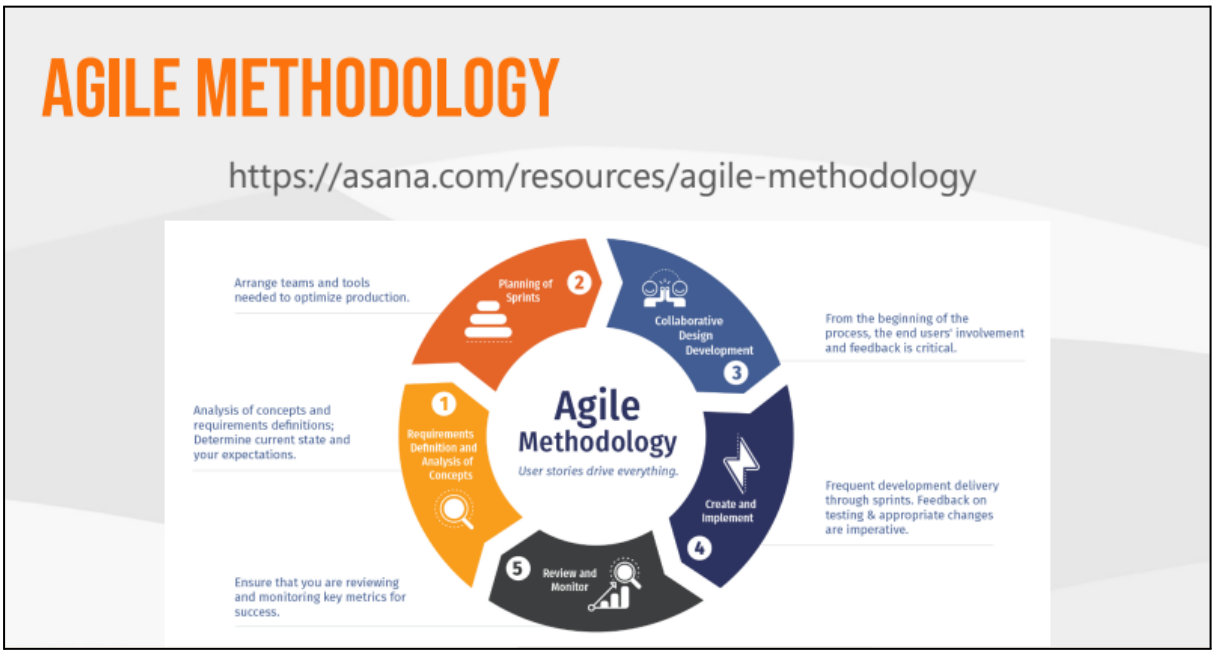
Encapsulation

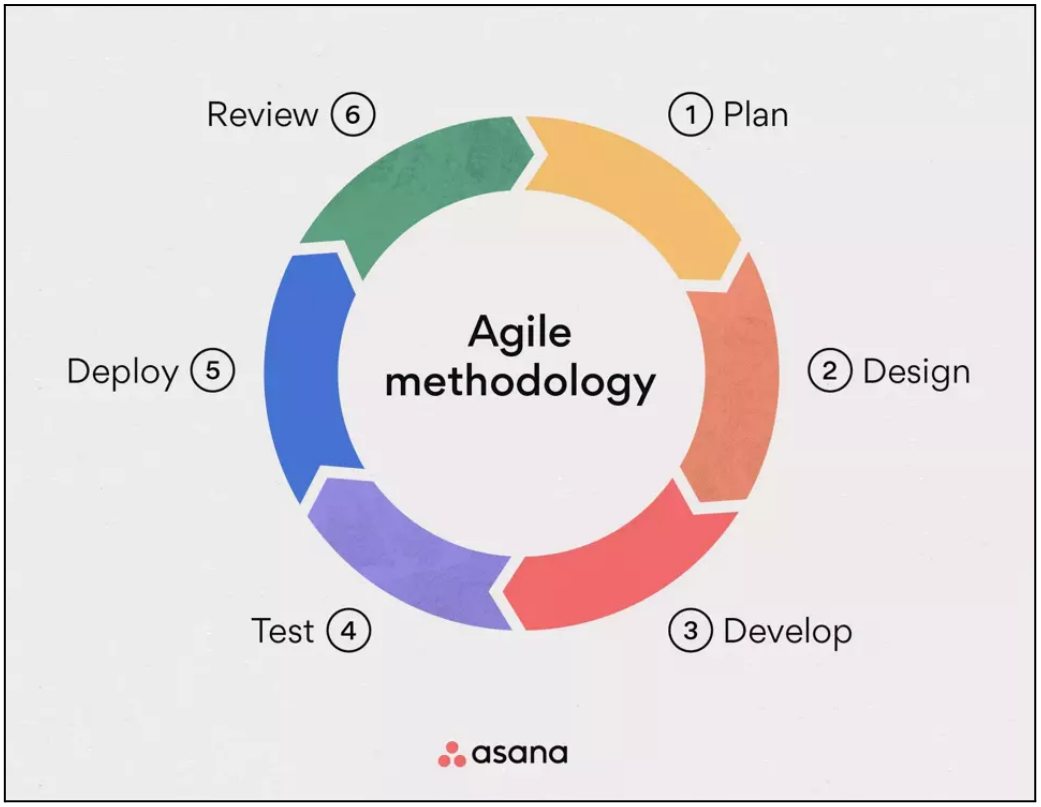
Is a process of **writing code and data together into a single unit**. We can create a fully encapsulated class in Java by making all the data members of the class private. We can use the setter and getter methods to set and get the data in it.

Agile Methodology

A project management framework that breaks projects down into several dynamic phases, commonly known as sprints. In this article, get a high-level overview of Agile project management, plus a few common frameworks to choose the right one for your team.

https://asana.com/resources/agile-methodology





LABORATORY 3

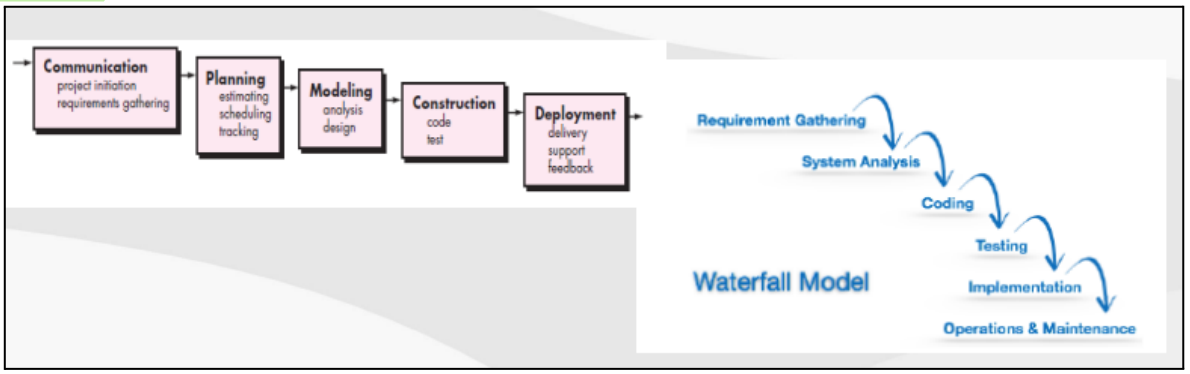
Review nyo yung Lab 3 na binigay nya, yung output giving emeru. Di ko na para ilagay yon dito matuto kayong tumayo sa sarili nyong paa.

https://drive.google.com/file/d/1pAOBqoPCFQGmhluc_clBywWi3m7ywxPl/view
https://drive.google.com/file/d/1nhShMpn1Sb5V4ZUMYg293Q3sQlpj0lmF/view
https://drive.google.com/file/d/1yj3NLIdKx-qSgbdzg22NZ2GoNLcpab2g/view

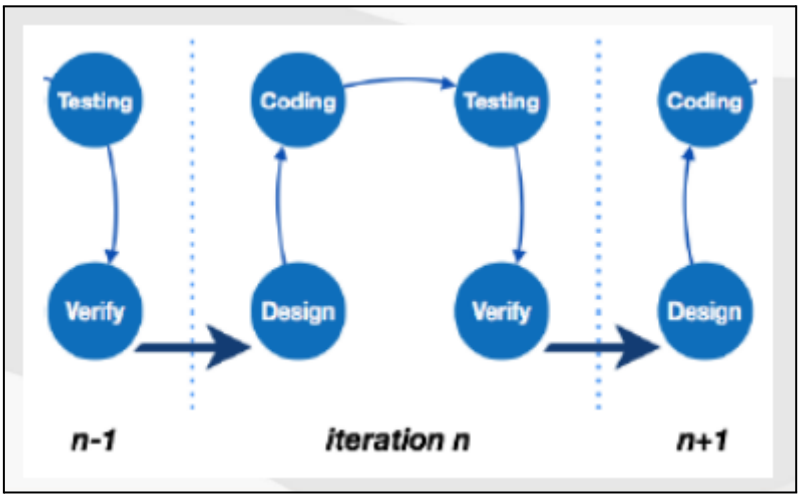
Link ulit.

ILLUSTRATIONS

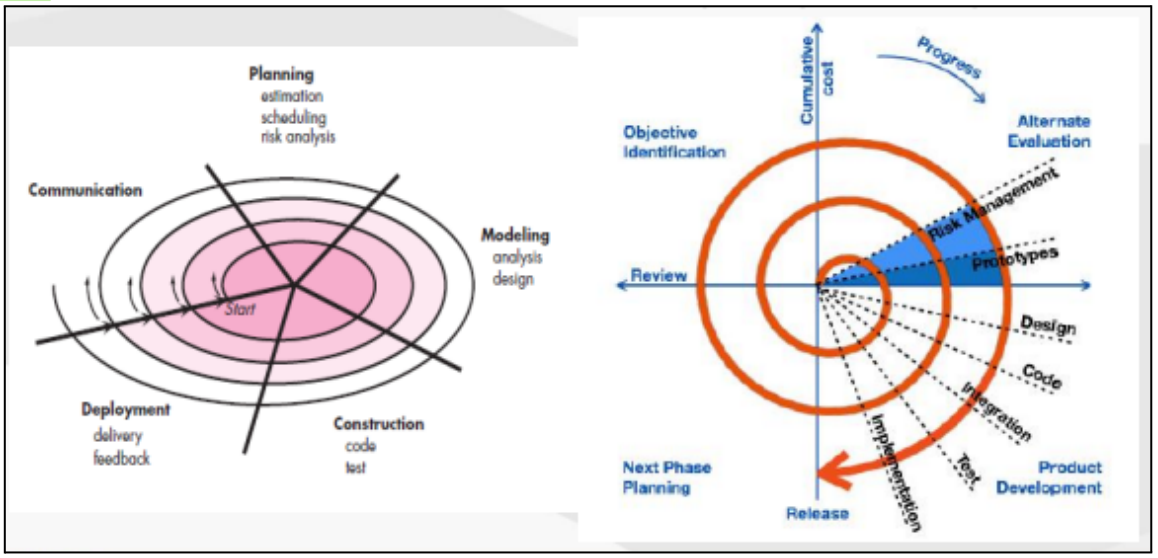
Waterfall



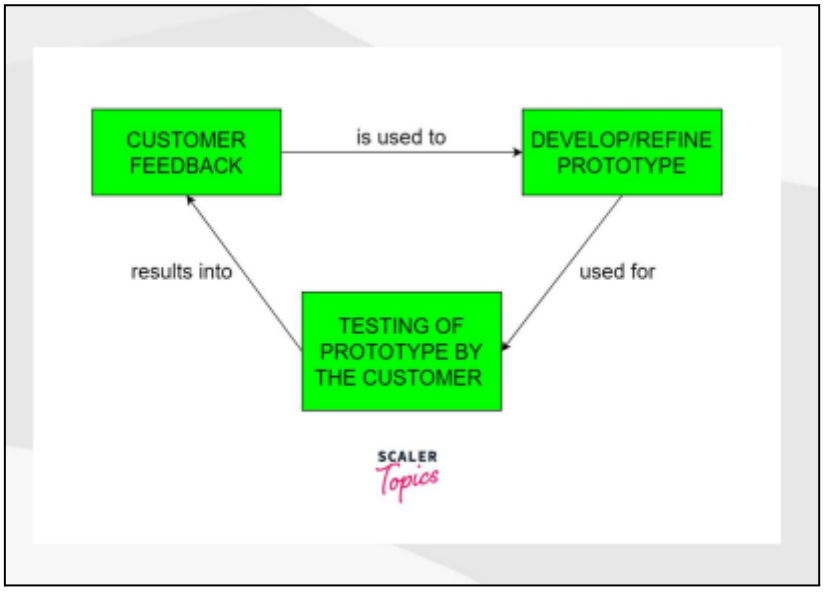
Iterative



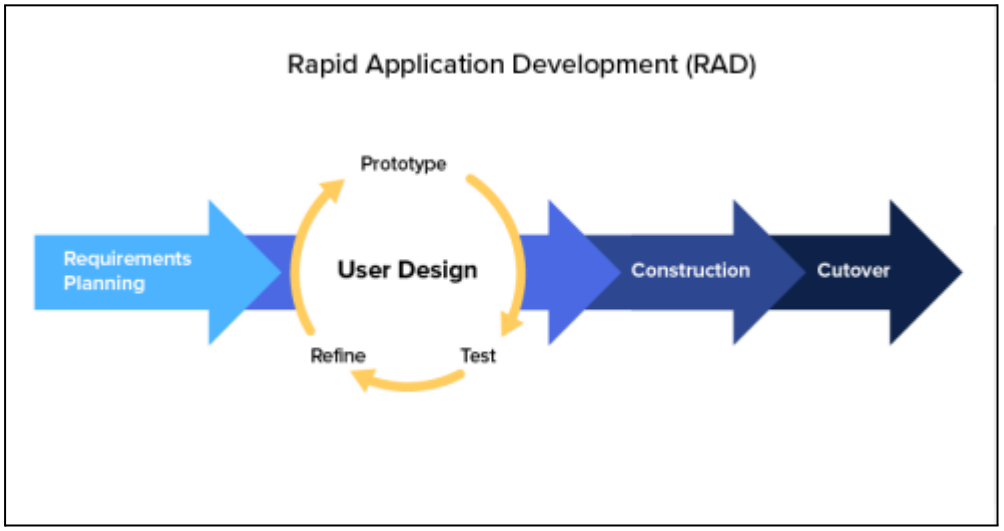
Spiral



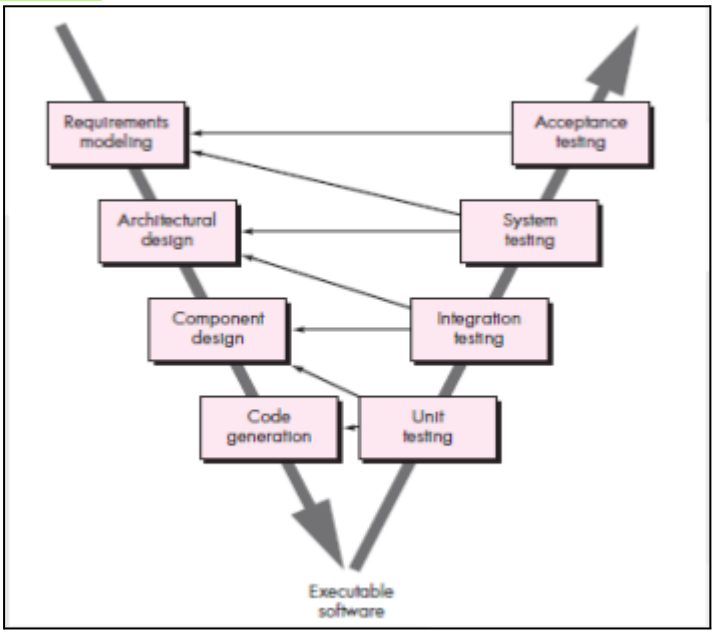
Prototyping



RAD


Rapid Application Development (RAD)
Prototype
Requirements Planning
User Design
Construction
Cutover
Refine
Test

V-Model


Requirements modeling
Acceptance testing
Architectural design
System testing
Component design
Integration testing
Code generation
Unit testing
Executable software