CSE 470 Software Engineering

Introduction - SDLC

Imran Zahid
Lecturer
Computer Science and Engineering, BRAC University

Project

Project Details

Project Purpose

The purpose of the project is to provide hands-on training on how to develop a software from scratch.

Key Definitions

Requirement: An action/capability that is expected of a software/product to fulfil

Feature: A set of logical actions that need to be performed in order to fulfil a part of a requirement

Project Details

Example of a requirement:

1. Customers can add products to their cart

Example of features:

For requirement (1), the features which need to be implemented are:

1. The system should be able to display products to customers
2. The system should be able to book products and mark them as added to the cart of a customer if the product is in stock

Project Details

Tech stack: The language and the environment which is used to develop and run an application.

Example of tech stacks

LAMP Stack (Linux OS, Apache web server, MySQL database, PHP language)

MERN Stack (MongoDB database, Express JS backend, React JS frontend, Node JS environment)

Project Constraints

- The project must have at least five (5) requirements with four (4) features each (at least one requirement must be conceptually unique).
- You may develop either a web application, android application, iOS application or a desktop application.
- You may use any tech stack/programming language/scripting to develop their application.
- The application developed MUST follow the MVC architecture.

Explanations: MVC Architecture

Software Architecture

It's like the design for a building; it outlines how different parts of software will work together.

Model-View-Controller (MVC) Architecture

MVC architecture separates an application into three components:

- The frontend (View), which presents information to the user
- The backend (Model & Controller), comprised of the Model which handles data and business logic; and the Controller, which acts as an intermediary, processing user input and updating the Model and View accordingly

Explanations: MVC Architecture

Example

Imagine you have a shopping app:

- The Model stores all the products and their prices.
- The View shows the products and lets you add them to your cart.
- The Controller takes your order, updates the inventory in the Model, and shows your updated cart in the View.

Explanations: MVC Architecture

- Now, suppose the database changes from MySQL to PostgreSQL. Should the Controller care at ALL?
  - No. The model should encapsulate that logic and only provide a method to the Controller that provides data.
- Again, suppose the internal logic of a decision changes in the Controller. Should the View have any kind of modification?
  - No. The View will just call the same method it always does, and whatever logic is necessary to be executed, the Controller will handle it.

Project Constraints

- Students may use existing frameworks (Laravel, Flask, Express, Angular, Vue) to help with the development as long as it implements the MVC architecture. (Django is not permitted since it implements MVVT instead of MVC architecture)
- Students may use libraries/packages to help with the development of their project as long as it does not implement any major feature of the project.

Project Constraints

- Students must develop the project from scratch. They may not take help from any source, copy code from any source or use any tutorials which directly shows how to implement the project that they have chosen.
- Students may not use a code base which they/others may have worked on before.
- Login/registration will not be considered as features or requirement for the purpose of this task
- Students must define the requirements and the features while submitting their project idea

Project Process Details

- We will follow Agile methodology to implement the project whereas scrum must be followed by the team. (will be explained below)
- The sprint durations will be of 2 weeks each, after the end of a sprint the sprint lead will be changed to another member of the group

Explanations: Agile Methodology

- Agile is a way of working on projects in small, repeatable steps. Instead of trying to complete everything at once, the team works on small parts of the project in cycles called "iterations" or "sprints."
- Each iteration should result in a usable piece of the project that can be shown to customers or stakeholders.
- Work is divided into short time frames, usually 1-4 weeks long, called sprints. At the end of each sprint, the team reviews what they've accomplished and plans the next steps.
- After each sprint, the team reflects on what went well and what could be improved, making adjustments for future sprints.

Project Process Details

- The sprint lead will have to collect the information from each of the group members about their project progress and submit it at the end of the sprint
- The information that the sprint lead needs to collect and/or provide are:
  - The tasks that each member wanted to complete during the sprint (this information has to be collected at the beginning of the sprint)
  - The tasks that each member has completed (this information has to be collected at the end of the sprint)
  - **Score each team member** according to the progress that they have made in completing their project
  - Identify and report the problems faced by each team member

Project Process Details

- Each group must use github to maintain their project code. There will be one repository per group and all the members should work there as a collaborator by creating branches.
- Each student must push their code to the project repo at the end of each sprint regardless of if there are incomplete features or errors or bugs.

Project Process Details

- After the sprints, each student must attend a viva where
  - They will show their project demo
  - They will be asked questions about their project code individually
- All viva will be recorded and stored for review purposes if possible
- The viva may be taken offline or online and the dates and times will be assigned by the section faculty

Project Process Details

- You MUST create the github repository for your project and add the section faculty as a collaborator to the repository during the first sprint
- You MUST submit your project idea, requirements and feature before the beginning of the first sprint
- I may change the project, the requirements and/or the features if the I determine that the project is not complex enough, or too complex or for purposes of removing potential duplication

Current Workload

- Start forming groups
  - A medium of communication will be provided, Discord or Slack, where you can find group members.
  - I will share a Google form soon, where you will submit the (maybe partial) list of group members.
  - I will merge incomplete groups at my discretion.
- Start thinking about project ideas
  - You will have to submit a project proposal before the first sprint, where you will have to mention all requirements and features. I will provide a template for the project proposal.

Fundamentals of Software Engineering

Fundamental Software Engineering Activities

- Communication
  - Requirement Collection
- Planning
  - Specification
- Modeling
  - Requirement Analysis
  - Design
- Construction
  - Code generation
  - Testing
- Deployment
  - Release
  - Maintenance

Fundamental Software Engineering Activities

- Specification:
  - Develop a detailed specification for the software
- Requirements analysis:
  - Analyze software system requirements in detail
- Design:
  - Develop detailed design for the software data structures, software architecture details, interfaces
- Coding:
  - Transform design into one or more programming language(s)

Fundamental Software Engineering Activities

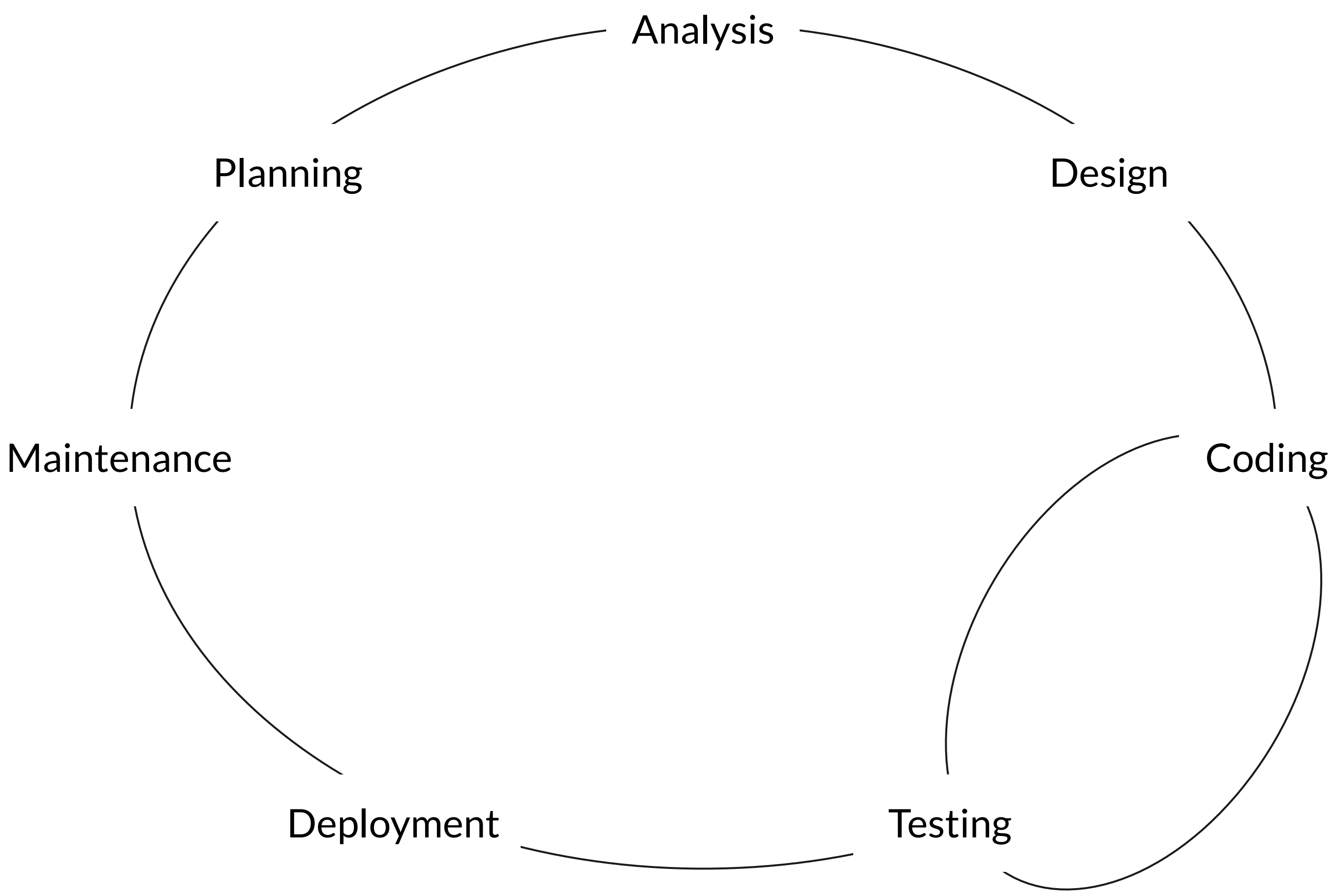
- Testing:
  - Test internal operation of the system and externally visible operations & performance
- Release:
  - Package and deliver software to users
- Maintenance:
  - Error correction and enhancement after system

Software Development Lifecycle

Software Development Lifecycle

- We discussed the Software Engineering tasks, but how does the software pass from one of these tasks to another?
- Software Development Lifecycle (SDLC) structured process that is used to design, develop, and test good-quality software.
- The life cycle defines a method for improving the quality of software and the all-around development process.

Why is it termed Lifecycle?



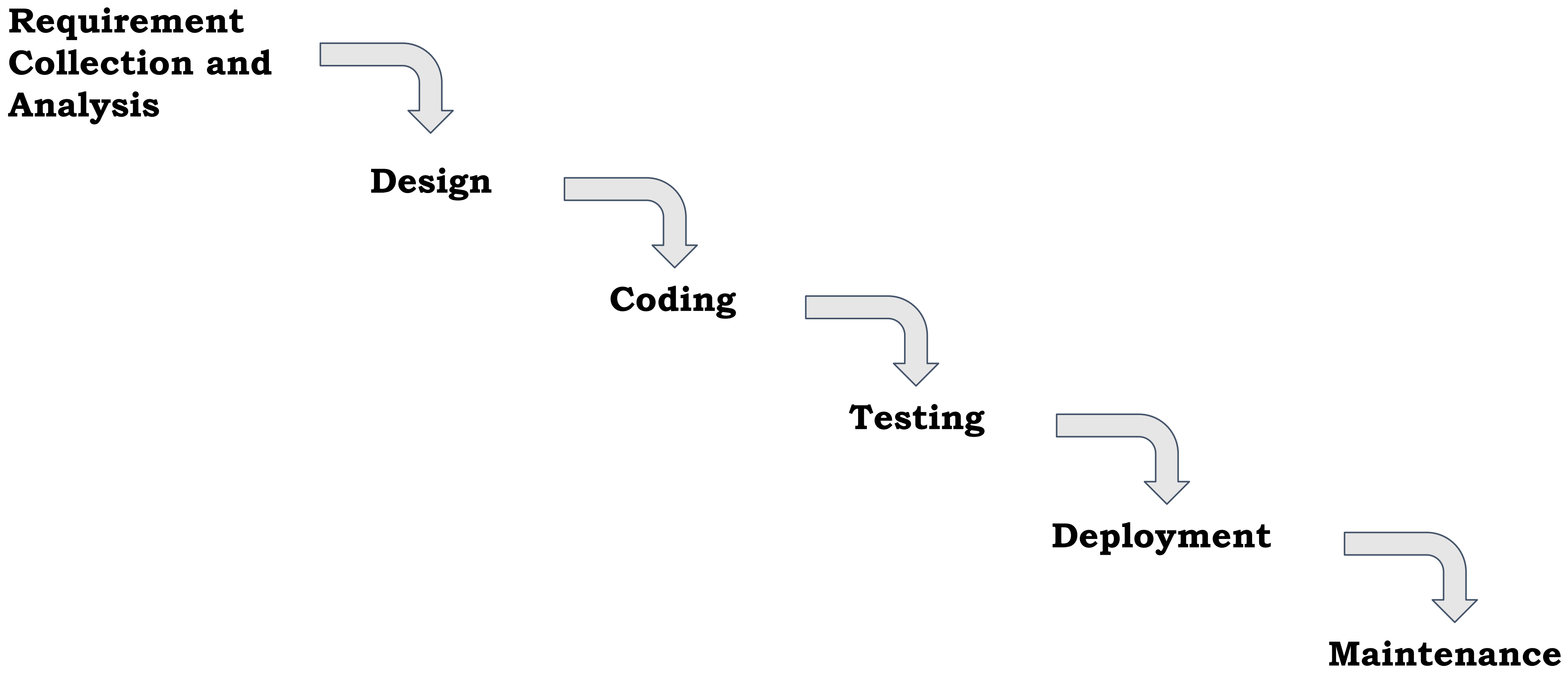
Is there only one possible lifecycle?

- No
- There isn’t a fixed flow of a program from one phase to another.
- One model may choose to have one final design, and then pass it to development.
- Another model may choose to revise the design depending on the feedback from developers.

So how do you choose an SDLC?

- Suppose your team lead has gotten a new project.
- How does he decide which SDLC to use? It depends on -
  - The scale of the software (e.g. how complicated will it be)
  - The size of the user base
  - How well the client understands what they want
  - How clearly the client can convey what they want

The most basic SDLC - Waterfall Model



Waterfall Model

- Suppose you have a very small scale project,
- And your client has clearly conveyed what he wants as the end product and there will be NO changes.
- In that case, one doesn't need a particularly complicated model.
- One such model is the Waterfall Model.
- Waterfall model can't accommodate any modifications to previous steps, either from testing or feedback.

Why study the Waterfall Model?

- Since there is no interaction between stages other than passing the outputs of a stage, it allows us to explore the stages in isolation.
- As you will see, it is the basic building block of other models, most other models will be some modification of the Waterfall model.

Requirement Collection

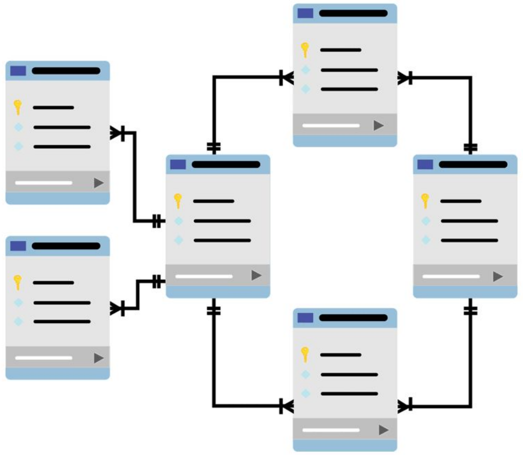
- Client Meeting
  - The process begins with discussing the customer's idea or project concept to understand their goals.
- Collect Requirements
  - Gather detailed information about what the customer needs and expects from the software.

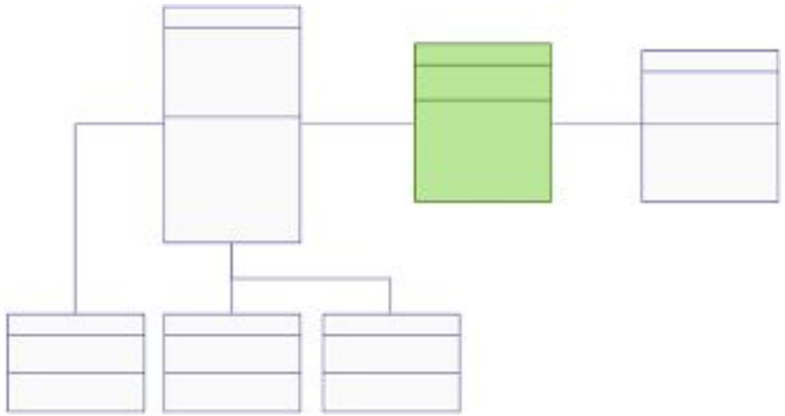
Requirement Analysis

- Address the Problem
  - Understand the core issue that the software needs to solve for the customer.
- Identify Feasible and Infeasible Requirements
  - Determine which requirements can realistically be implemented and which cannot due to constraints.
- Define How the Software Will Meet Customer Requirements
  - Establish a plan for how the software will fulfill the customer's needs and expectations.

Design

- Create the logical and physical design of the software project.





Coding

- Build the Software
  - Construct the actual software based on the design.
  - Development cannot begin until the design is thoroughly finalized.
- Convert Design into Functional Software
  - Begin turning the design into working software by writing code.
- Break Design into Code Modules
  - The design is divided into smaller parts, and each part is developed as a code module, one at a time.

Testing

- Validate Software Against Requirements
  - Ensure the software meets the requirements identified during the analysis phase.
- Fix Issues as They Arise
  - If any problems are found, make corrections directly in the code to resolve them.

Deployment and Maintenance

- Deploy the Software in the User Environment
  - Release the software for use in the actual information technology environment where users will operate it.
- Perform Deployment and Maintenance Gradually
  - Roll out the software step by step and carry out ongoing maintenance.
- Address Issues Promptly
  - Any problems that arise are handled and resolved by the deployment team.
- Collect User Feedback
  - Gather input from users to improve the software or address any concerns.

When to choose Waterfall Model

- Requirements are well known
- Small scale and short term project
- Resources are available and trained
- Technological tools required are not dynamic, instead are stable

Pros

- Simple to Use and Easy
- Stages go one by one, so sudden changes can not create confusions
- Any changes are done only in development stage, so no need to get back and change everything.

Cons

- While completing a stage, it freezes all the subsequent stages.
- No way to verify the design
- Once in testing phase, no more features can be added
- Code reuse not possible

Thank you