Software engineering

Professor Sabina Umirzakova

Introducing yourself

- 1. Name
- 2. Which computer language do you know?
- 3. Your programme skills from 1 to 5 ?

Grading

Attendance	10%
Midterm Exam	20%
Assignments(presentation)	30%
Final exam	40%

Homework and Programming projects will be posted online on the class webpage

Those who miss 4 classes will be awarded F

This course focus

- **Software Engineering** course delivers basic and advanced concepts of Software Engineering.
- Software Engineering course is designed to help beginners and professionals both.
- Software Engineering provides a standard procedure to **design and develop a software.**

This course focus

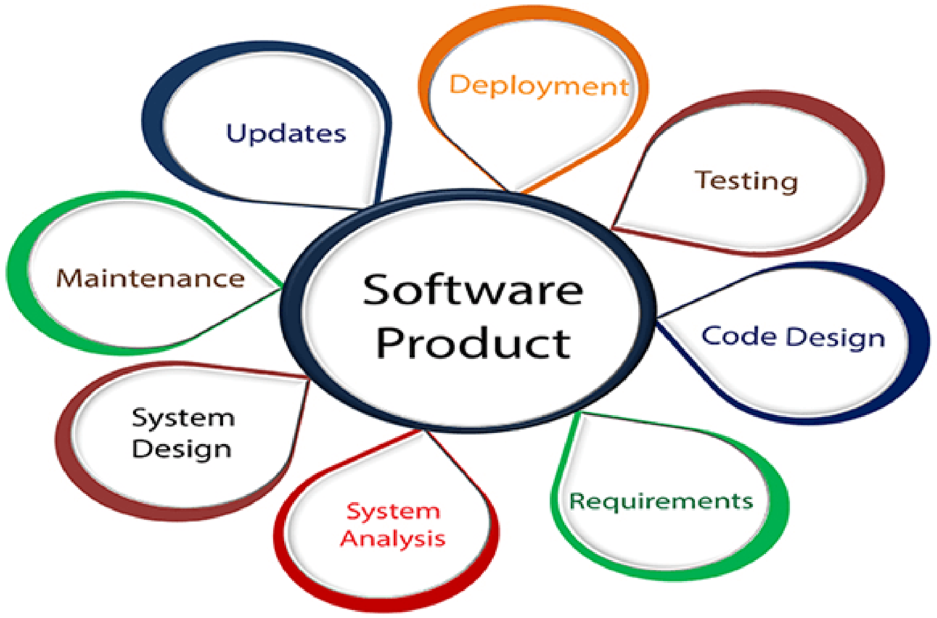
- This course will focus how to work with projects !!!!!!!!!!!!
- For project you can use any **program language**.
- In this course will be many **assignment (every class).**

What is Software Engineering

- The term **software engineering** is the product of two words, **software**, and **engineering**.
- The **software** is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various **particular computer languages.**
- Computer programs and related documentation such as requirements, design models and user manuals.
- **Engineering** is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain**, and **improve frameworks, processes, etc**. Aka tools and methods to find effective solution to problems.

What is Software Engineering

- **Software Engineering** is an engineering branch related to the evolution of software product using **well-defined scientific principles, techniques, and procedures**.
- The result of software engineering is an effective and reliable software product.



Why is Software Engineering required

Software Engineering is required due to the following reasons:

- To manage Large software
- For more Scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

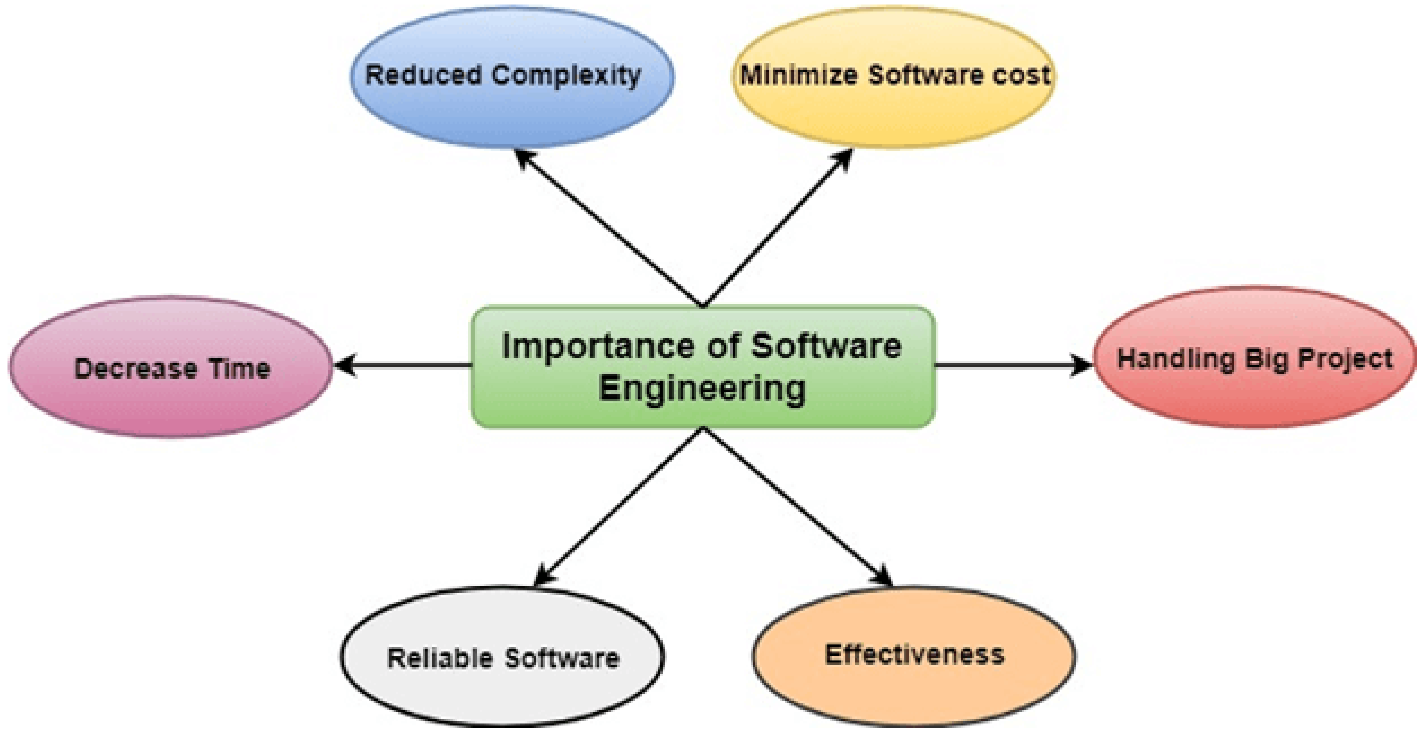
Need of Software Engineering

- **Huge Programming:** It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.
- **Adaptability:** If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.
- **Cost:** As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware. But the cost of programming remains high if the proper process is not adapted.
- **Dynamic Nature:** The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.
- **Quality Management:** Better procedure of software development provides a better and quality software product.

Characteristics of a good software engineer

- **The features that good software engineers should possess are as follows:**
- Exposure to systematic methods, i.e., familiarity with software engineering principles.
- Good technical knowledge of the project range (Domain knowledge).
- *Good programming abilities*.
- Good communication skills. These skills comprise of **oral, written, and interpersonal skills.**
- High motivation.
- ETC…

Importance of Software Engineering



Software Processes

- The term **software** specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) **that describe the program and how they are to be used**.
- These are four key process activities, which are common to all software processes:

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software development:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

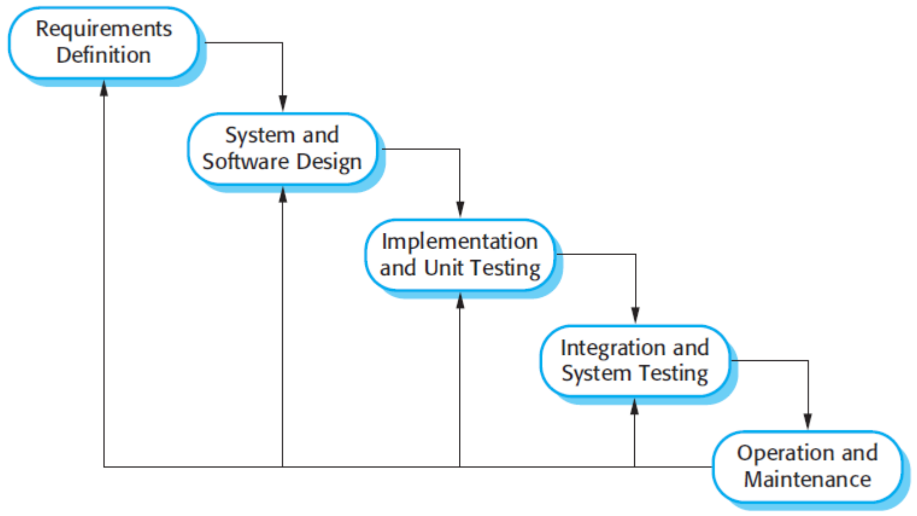
The Software Process Model

- A software process model is a specified definition of a software process, which is presented from a particular perspective.
- Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described.
- Process models may contain **activities**, which are part of the software process, **software product,** and the **roles of people** involved in software engineering.

The Software Process Model

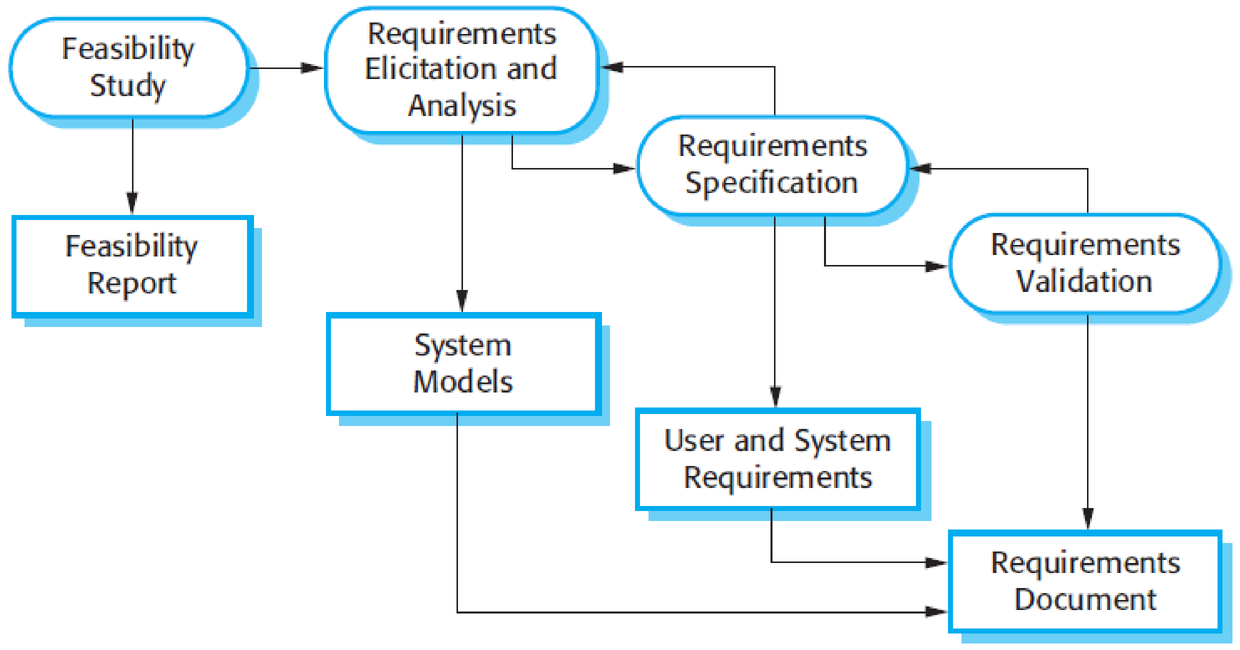
- Some examples of the types of software process models that may be produced are:

1. In software engineering, a "**workflow model**" refers to a visual representation of the sequential steps and decision points involved in a software development process, outlining the tasks, dependencies, and flow of information from the initial concept to the final product delivery, essentially acting as a roadmap for a project to ensure efficient development and organization.


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

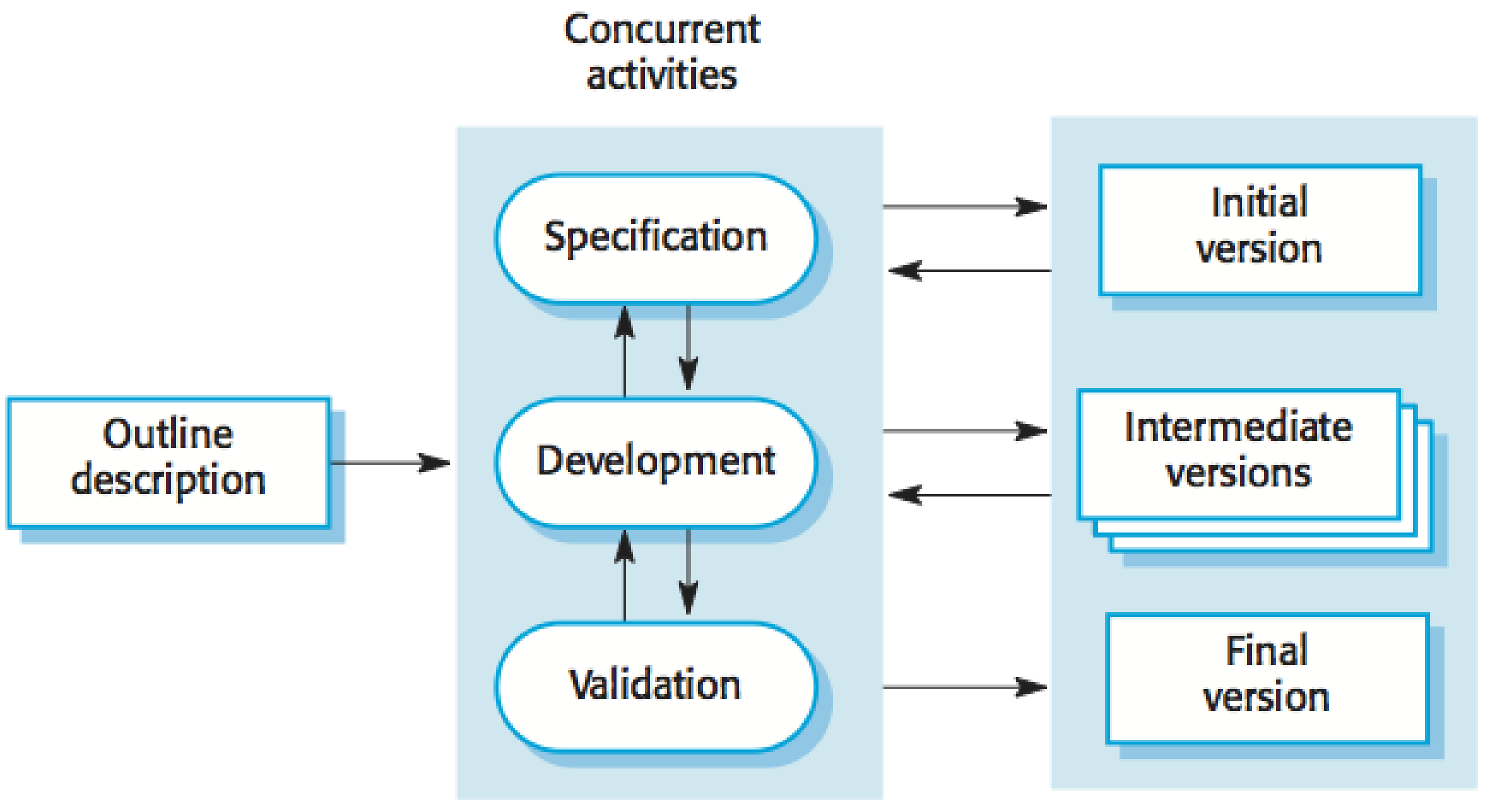
The Software Process Model

- **2. A dataflow or activity model.** This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design.



The Software Process Model

- **A role/action model.** This means the roles of the people involved in the software process and the activities for which they are responsible.



Software Crisis

- **Software crisis** it is a term describing difficulties of writing **useful** and **efficient** program in required time.

Software Crisis

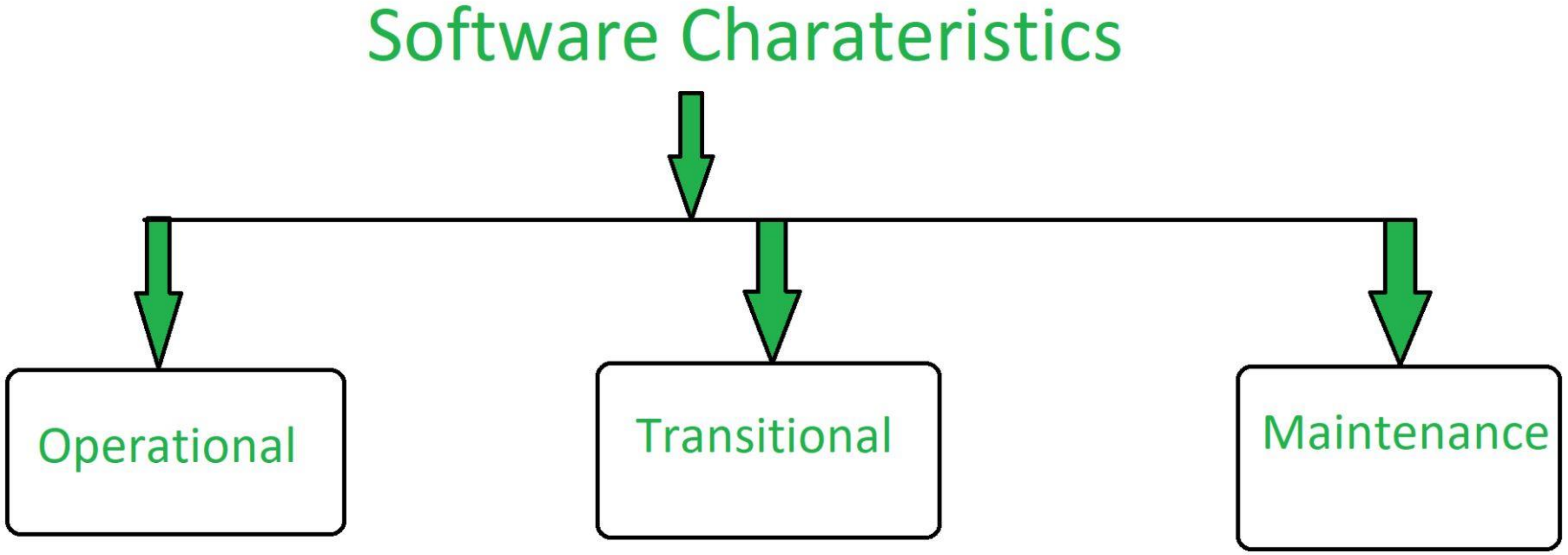
1. **Size:** Software is becoming more expensive and more complex with the growing complexity and expectation out of software. For example, the code in the consumer product is **doubling every couple of years**.
2. **Quality:** Many software products have **poor quality**, i.e., the software products defects after putting into use due to ineffective testing technique. For example, Software testing typically finds 25 errors per 1000 lines of code.
3. **Cost:** Software development is costly i.e. **in terms of time** taken to develop and the money involved. For example, Development of the FAA's Advanced Automation System cost over $700 per lines of code.
4. **Delayed Delivery:** Serious schedule overruns are common. Very often the software takes longer than the estimated time to develop, which in turn leads to cost shooting up. For example, one in four large-scale development projects is never completed.

Program vs. Software

- Software is more than programs. Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared. There are three components of the software:
- **1. Program:** Program is a combination of **source code & object code**.
- **2. Documentation:** Documentation consists of different types of manuals. Examples of documentation manuals are: **Data Flow Diagram, Flow Charts, ER diagrams, etc.**
- **3. Operating Procedures:** Operating Procedures consist of instructions to set up and use the software system and instructions on how react to the **system failure**. Example of operating system procedures manuals is: installation guide, Beginner's guide, reference guide, system administration guide, etc.

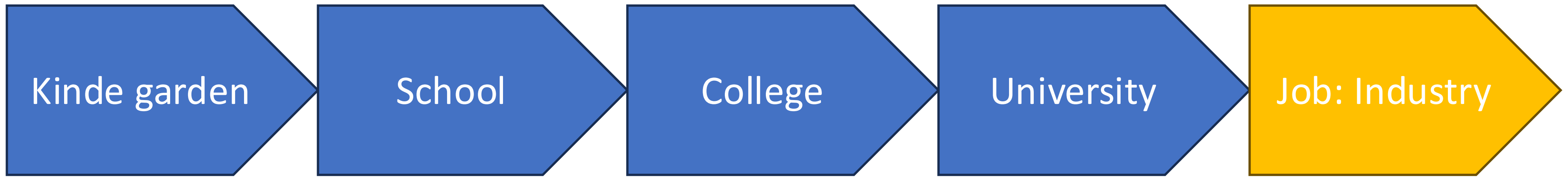
Characteristics of good software

- A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:



Software Development Life Cycle (SDLC)

- A **software life cycle model is a sequence of work to final output (results)**



Software Development Life Cycle (SDLC)

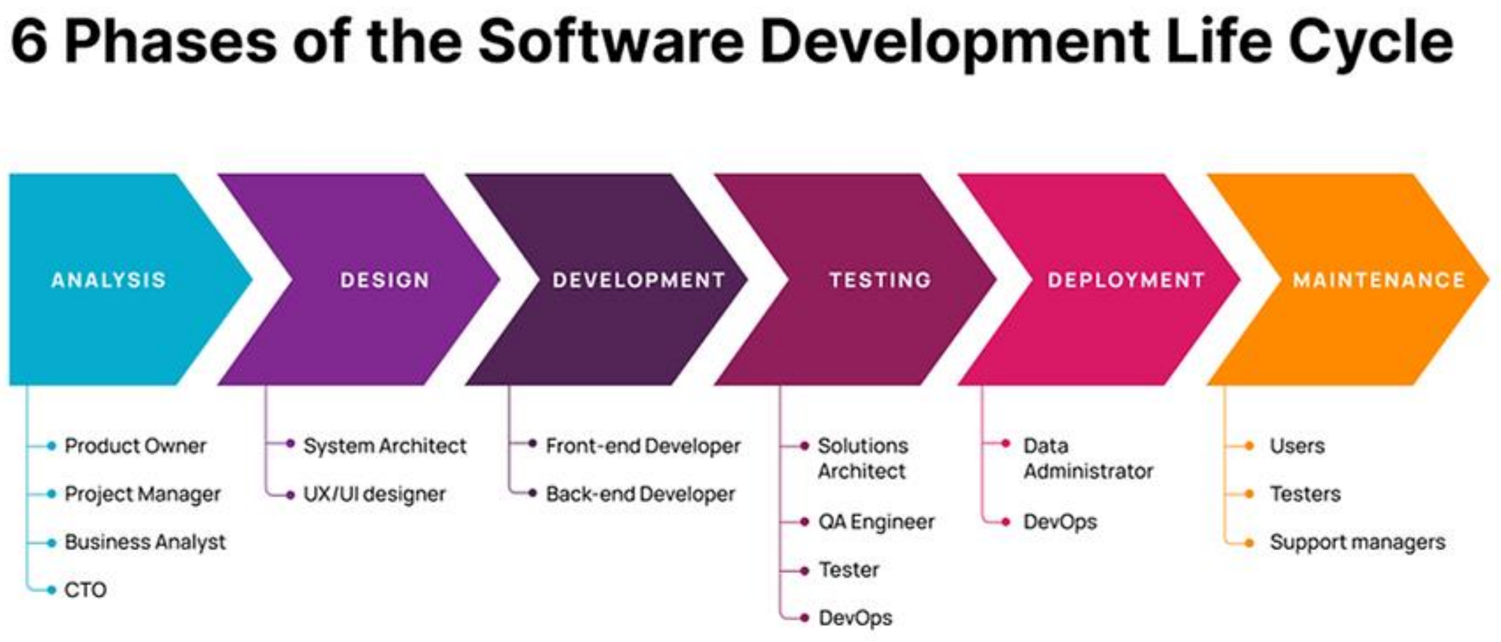
- A **software life cycle model** (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its **life cycle stages (step by step)**.
- A **life cycle model** maps the various activities performed on a software product from its inception to retirement.
- Different life cycle models may plan the necessary development activities to phases in different ways.
- During any life cycle stage, more than one activity may also be carried out.

Need of SDLC

- The development team must determine a suitable life cycle model for a particular plan and then observe to it.
- Without using an **exact life cycle model**, the development of a software product would not be in a systematic and disciplined manner.
- When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, it **would point to chaos and project failure**.

SDLC Cycle

- SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



The stages of SDLC: **Planning and requirement analysis**

- **Requirement Analysis** is the most important and necessary stage in SDLC.
- **Planning** for the quality assurance **requirements** and identifications of the **risks** associated with the projects is also done at this stage.
- Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product.

The stages of SDLC: **Defining Requirements**

- Once the requirement analysis is done, the next stage is to certainly **represent and document the software requirements** and get them accepted from the project stakeholders.
- This is accomplished through "SRS"**- Software Requirement Specification document which contains** all the product requirements to be constructed and developed during the project life cycle.

The stages of SDLC: **Designing the Software**

- The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project.
- This phase is the product of the last two, like inputs from the customer **and requirement gathering.**

The stages of SDLC: **Developing the project**

- In this phase of SDLC, the actual development begins, and the programming is built.
- The implementation of **design begins** concerning **writing code.**
- Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and **implement the code**.

The stages of SDLC: **Testing**

- After the code is generated, **it is tested against the requirements** to make sure that the products are solving the needs addressed and gathered **during the requirements stage**.
- During this stage, **unit testing, integration testing, system testing, acceptance testing are done**.

The stages of SDLC: **Deployment**

- Once the software is certified, and no bugs or errors are stated, then it is deployed.
- Then based on the assessment, the software may be released as it is or **with suggested enhancement in the object segment**.
- After the software is deployed, then its maintenance begins.

The stages of SDLC: **Maintenance**

- Once when the client starts using the developed systems, then the real issues come up and requirements to be solved **from time to time**.
- This procedure where the care is taken for the developed product is known as maintenance.

Homework

- Math Exercises. Calculator
- Math exercises are a good place to get comfortable with Python’s syntax and style. Write some code to conduct simple math calculations (addition, subtraction, etc.) but with caveats.
- For instance, write a short program that asks the computer to request and add two variable inputs. Then, adjust the program to perform a specific task based on the result. For example, print a specific statement if the sum is between 10 and 20, above 100, or print “no negative numbers” if a subtraction problem delivers a corresponding result. You can gradually expand the program to make it more elaborate and the tasks more complex.

Homework

- 1. Prepare program based on description.
- 2. Prepare detailed presentation of how your code is working.
- 3. Explanation on desc 5~10 minutes.