Slide 02 : Software Development Life Cycle

Made by Md. Mehedi Hasan Rafy

Software Development Life Cycle (SDLC)

A software life cycle model is a structured, descriptive and diagrammatic representation showing all activities from software inception to retirement.

It defines phases, order of execution, and entry/exit criteria.

Need a Life Cycle Model

- Ensures systematic and disciplined development.
- Creates a clear understanding among team members about what and when to do tasks.
- Defines phase entry/exit criteria, so work proceeds in a controlled way.
- Enables project managers to monitor progress effectively.

Common Life Cycle Models

1. Classical Waterfall Model
2. Iterative Waterfall Model
3. Prototyping Model
4. Evolutionary Model
5. Spiral Model
6. RAD Model
7. Agile Model

1. *Classical Waterfall Model*

Classical Waterfall Model is the most earliest and simplest model. It is very structured and moves step-by-step from start to finish.
It is more theoretical than practical, but it is the basis for other models.

Phases of Classical Waterfall Model

1. Feasibility Study
2. Requirements Analysis and Specification
3. Design
4. Coding and Unit Testing
5. Integration and System Testing
6. Maintenance

1. ***Feasibility Study***

   - ***Goal:*** Decide if the project is financially and technically feasible.
   - ***Steps:***
     - Project manager or team leaders try to understand the client's needs.
     - Study input/output data, processing requirements, and various constraints for the system.
     - Evaluate possible solutions in terms of cost, resources, time, and technical expertise.
     - Select the best feasible solution.

2. ***Requirements Analysis & Specification***

   - ***Goal:*** Understand exact customer requirements and document them in the SRS (Software Requirements Specification).
   - ***Steps:*** This phase consists of two distinct activities:
     - ***Requirements gathering and analysis***
       - Gather relevant data from users and clients through interviews and discussions.
       - Identify and resolve contradictions and ambiguities.
     - ***Requirements specification:***
       - User requirements identified in the requirements gathering and analysis activity are systematically organized into a SRS document.
       - Important components of SRS documents are functional requirements, non-functional requirements, and implementation goals.

3. ***Design***

   - ***Goal:*** Derive a software architecture from the SRS document.
   - ***Two approaches****:*
     - ***Traditional Design***

       Traditional design consists of two different activities.

- A structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined.
- After structure analysis, structured design activities are started. During this activity, the results of structured analysis are transformed into the software design.

- ***Object-Oriented Design***
  - Various objects and relationships among these objects that exist in the problem domain and the solution domain are identified.
  - The object is further refined to obtain detailed deign.

4. ***Coding & Unit Testing***

   - ***Coding:*** Translate the software design into source code. The design is implemented as program modules and the end-product is a set of program modules that have been unit tested.
   - ***Unit Test:*** Unit testing refers to testing each module of the end-product in isolation to determine correct working of all of the individual modules. It is the most efficient way to debug the errors identified at coding stage

5. **Integration & System Testing**

   - ***Integration:*** After coding and unit testing stage, modules are integrated incrementally over a number of steps. During each incremental step, the partially integrated system is tested and a set of previously planned modules are added to it.
   - ***System Testing:*** After all the modules have been successfully integrated and tested, system testing is carried out.

     Types of system testing activities are:

     - ***α – Testing:*** It is the system testing performed by the development team.
     - **β – Testing:** It is the system testing performed by friendly set of users.
     - **Acceptance Testing** – It is the system testing by the actual customer after the product delivery to determine whether to accept or reject the delivered product.

6. **Maintenance**

   - Maintenance involves performing one or more following activities:

- **Corrective** – Fix errors that were not discovered during product development phase.
- **Perfective** – Improve implementation or add new features according to customer's requirements.
- **Adaptive** – Port to new platforms or environments like to a new operating system.

Shortcomings of Waterfall Model

- Assumes no errors in any phase – unrealistic.
- In reality, mistakes happens in practical development environments, due to wrong assumptions, oversight, communication gaps, or wrong technology.
- Many defects are only detected late in the project, making fixes costly.