DIPLOMA WALLAH

(Your One Stop Hub For Diploma Resources)

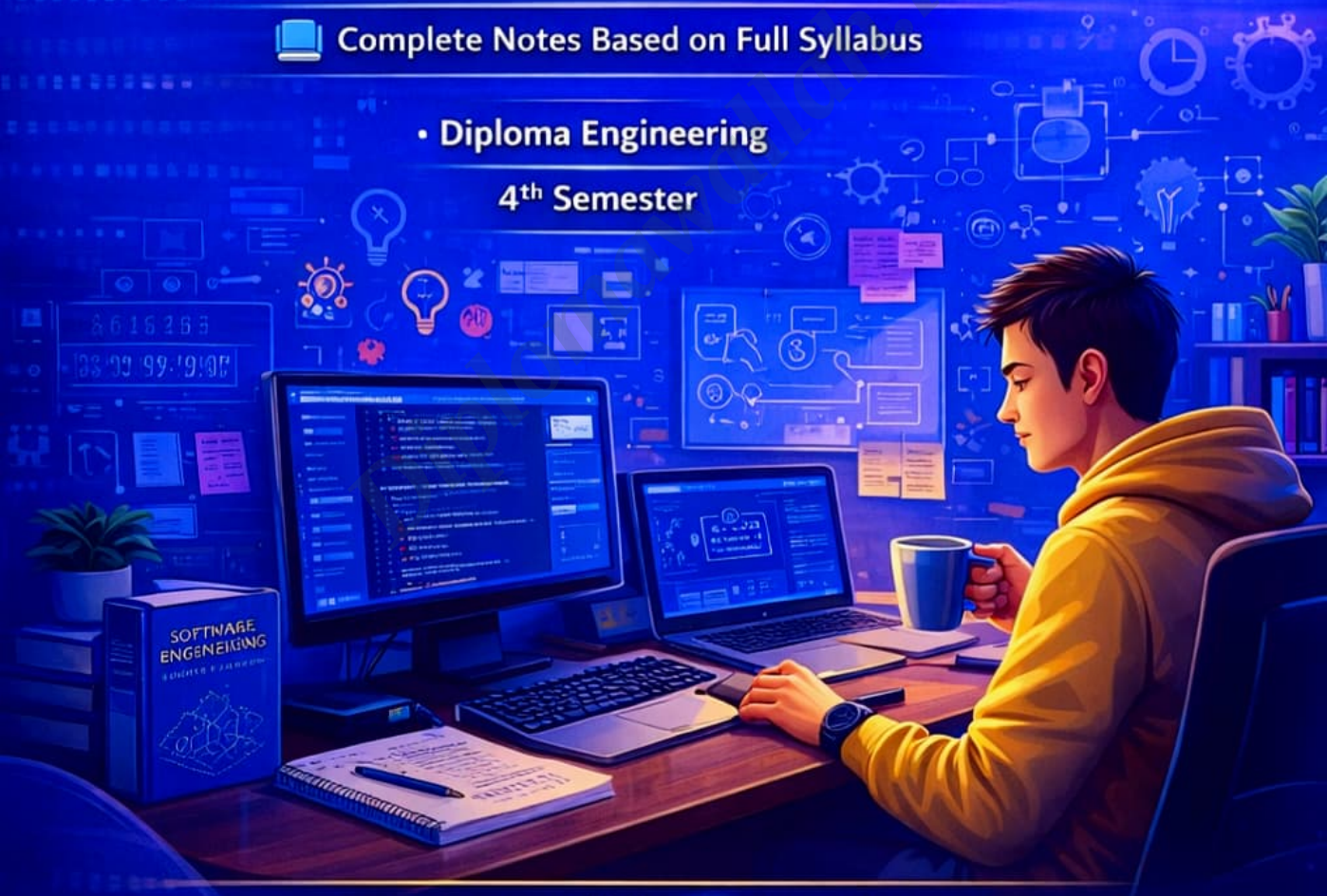
SOFTWARE ENGINEERING

PRINCIPLES AND PRACTICES

Complete Notes Based on Full Syllabus

- Diploma Engineering

4th Semester



www.diplomawallah.in

Notes prepared by **Sangam**

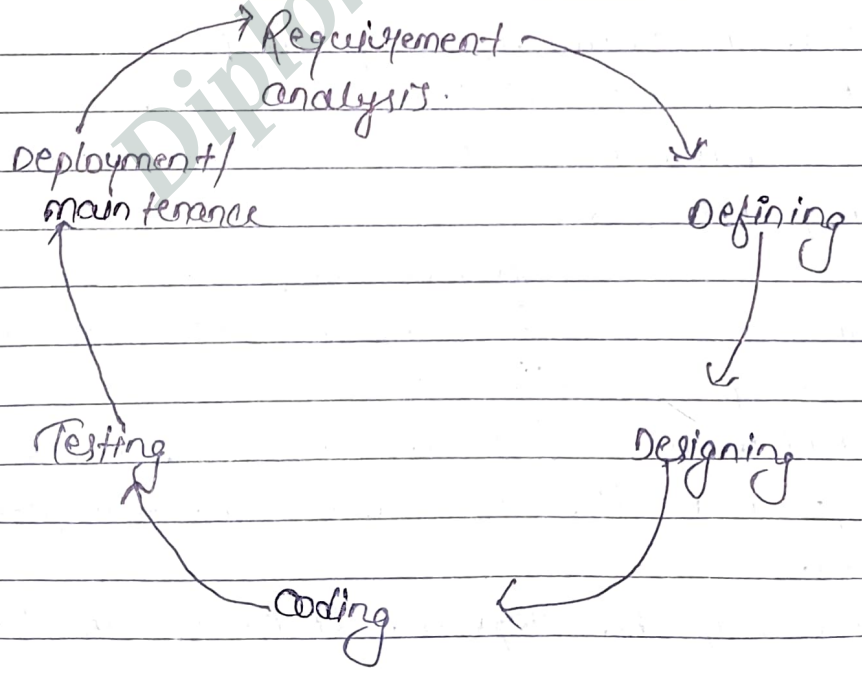
Unit-02

SDLC (Software development life cycle)

SDLC is a structured process that is used to design, develop and test high-quality software. Software development lifecycle is a methodology that defines the entire procedure of S/w development Step by Step.

The goal of the SDLC life cycle model is to deliver high-quality, maintainable S/w that meet the user requirement.

In other words, SDLC is a structured process followed for the development of S/w, from Initial requirement gathering to deployment and maintenance.



SRS Document

1. Requirement Analysis / planning :-
(product owner / project manager / Business Analyst / CTO) → chief technical officer.
- Gather analyze user requirements.
- Deliverable : SRS (Software Requirement Specification)

*purpose of Req. Analysis
→ To know what the S/w should do.
→ To avoid miscommunication b/w client and developers.
→ To plan the design, development and testing properly.
→ To reduce cost and rework later.

Steps:
1. Requirement Gathering
→ Talk to stakeholders, users, customers
→ use interviews, surveys, meetings.
2. Requirement Elicitation
→ Discover hidden needs or unclear requirements.
3. Req. Analysis
— Study and categorizes the req.
— Find out conflicts, duplicates or missing points.
4. Req. specification
— Document the final, clear and agreed-upon req. in a SRS (Software requirement Specification)
5. Req. Validation
— Check if the req. are correct, complete, and possible to implement.

O/p of Req. Analysis
- SRS Document (contains all functional & non-functional req. in written form, approved by both client and development team.)

Benefits
- Save time and money.
- Improves customer satisfaction.
- Reduce bugs and rework.
- makes testing and design easier.

Types of Req. Analysis
① Functional Req. (what the system should do)
Example:- • Student management
- faculty login & dashboard
- Attendance Entry
- Alerts & notification,
- Student portal

* Non-functional Req.
(How the System Should behave)
example:- System Should load within 2 Second.
- performance
- Security
- Data protection
- Scalability
- Backup & Recovery.

2. Defining

Defining means to clearly write down (document) all the analyzed requirement in an organised and structured way - usually in a SRS (Software Requirements Specification) document.

purpose :-

- Create a formal agreement b/w customer and developers.
- Guide for design, coding and testing.
- Ensure nothing is missed or misunderstood.

3. Designing :-

(System Architect / UX/UI designer)

Designing is the phase where we create a blueprint or plan for how the software system will work before coding begins.

It transforms the requirement (from SRS) into a technical solution that developers can build.

purpose :-

- To plan the structure of the software.
- To define how components will interact.
- To make sure the system is efficient, scalable and maintainable.
- To reduce errors during coding and speed up development.

Types of System design

1. Architectural design :- High level structure of the system. Defines main components and their relationship.
2. High-level design (HLD) :- Describe modules, interfaces, data flow - also called "System design."
3. Low-level design :- Detailed design for each module - logic, functions, class, diagrams, pseudocode etc.

Designing include :-

- System architecture :- Client-server, layered, etc
- Database design (tables, relationship)
- User interface design (screens, layout)
- Component design (modules, classes, functions)
- Security and performance consideration.

* Common Tools used :-

- DFD (data flow diagrams)
- ER diagrams (Entity Relationship Diagrams)
- UML (unified modeling language)
  - Class diagram
  - Sequence diagram
  - Use case diagram
- Flowcharts
- Wireframe (new design)
- Database design
- UI/UX

Benefits
- Reduces development and testing time.
- Improves code quality and maintainability.
- Enhances performance and scalability.

Examples:—
Requirement: 'User can register with email and password.'

In design:-
- Front-end screen with input fields.
- Back-end module for validation.
- Database table to store user info.
- Security rules for password storage.

Design → Low level design → High level design

Stage 03 Developing the project/product:

At this stage, the fundamental development of the product starts and the programming is built. For this developers use a specific programming code as per the design programming code as per the design in the DDS.

Sangam

* Coding (development)

Coding is the phase in software development where the actual source code is written. This phase comes after designing and involves translating the system and software design into executable computer programs using a programming language like python, java, C++ etc.

Ex-

→ We already have a clear design (blueprint) from the previous step.

→ Now, we write actual code to bring the design to life.

→ This is where developers / programmers come into play and type the logic using programming lang.

Activities in Coding phase

→ Choosing right programming lang.

→ Following coding standards and best practices.

→ Writing clean, understandable, and error-free code.

→ Unit Testing the code.

→ Ensuring the code is efficient and maintainable.

(This is the longest and most important phase.)

*Testing

Testing is the process of executing a program or system with the intention of finding errors.
It checks whether the actual software output matches the expected output and ensure the s/w is defect free.

purpose of Testing
→ To detect bugs or errors in the s/w.
→ To ensure the s/w works as expected under different conditions.
→ To verify that the software meets the specified requirements.
→ To improve the quality, performance, and security.
→ To ensure user satisfaction.

Type
+ unit Testing — Testing individual units or components of s/w.
+ Integration Testing — Checks if different modules or services work well together.
modules by module
System Testing — Tests the complete s/w system as a whole.
Acceptance Testing — Done by the end-user to verify the system meets business req.

depend on use of { saas / paas } → software (small) platform

* Deployment

Deployment is the process of delivering the final s/w product to the end user or client so that it can be used in the actual environment.

This is one of the final stages of the Software Development life cycle (SDLC).

* main steps in deployment

1. packaging the s/w:-
   - The s/w is compiled and packaged into an executable format (like .exe, .apk, .zip etc).

2. Installation:-
   - The software is installed on the client's system, server, or cloud platform (depending on the type of s/w).

3. Configuration:-
   - Setting up system settings, databases, API and other environment variables.

4. Training and user Support:
   - Sometimes user are trained to use the new system.
   - Help documentation and support may be also be provided.

5. Going live:- The software becomes active and available for actual use by users.

6. post-deployment Support:-
   After deployment, maintenance, bug fixing and regular updates are provided.

Types

* Manual Deployment :- done by developers / admins.
* Automated Deployment - done using tools like Jenkins, Github Actions etc.
* phased Deployment :- Released in phase to limited audience first. then to all.

Example :- Suppose you made a college management System. After coding and testing.
→ you install it on the college's server.
→ Configure it with real Student data.
→ Train staff to use it.
→ Then the system becomes live - this is deployment.

* Maintenance
Maintenance is the last and longest phase of the SDLC. once the s/w developed and deployed / delivered to customer), it may still need changes, fixes or improvements. This is where maintenance comes in.

Software maintenance means making changes to the s/w after it is delivered to the user, to fix bugs, improve performance, or adapt it to new environment.
* Why it needed?
→ Fixing errors, Improving performance, Adapting to new environment.

Software process model:-

1. Waterfall model
This is a linear and sequential model. Each phase must be completed before moving to the next.

A Software process model is a structured approach to s/w development that defines the steps to follow for designing, developing, testing and maintaining s/w.

Types:-
- Traditional process model.
- Agile process model.

* Traditional models
→ Follow a strict sequence of steps (planning first, coding later)
→ Best for large projects where requirements are fixed.
→ less flexibility - Changes are difficult to incorporate.

Types of Traditional Models:-
1. Waterfall model
2. Iterative waterfall model
3. prototyping model.

* Waterfall Model
A linear, step-by-step approach where the next phase begins only after the previous one is completed.

phase (development)
Requirement → Design → Implementation → Testing → Deployment → Maintenance

Feature
- Simple and easy to understand
- Best for project with well-defined req.
- No flexibility - changes are difficult.

Advantage
- Easy to manage & document
- Works well for small projects.

Disadvantage
→ No changes allowed once the process starts
→ Testing happen late, so errors are costly.

* Iterative waterfall model
An improved version of the waterfall model where feedback is allowed after each phase.

Features:-
→ more flexible than the basic waterfall model.
→ Can improve the software based on feedback.
→ Still follows a structured flow, making changes slow.

Advantage
- Error can be fixed in early stages.
- more structured than waterfall.

Disadvantage
- Can still be time consuming
- Not suitable for rapidly changing req.

Ex
Writing an essay - you write a draft, then revise multiple times to improve.

* use when req. not fully done.

* Prototyping Model

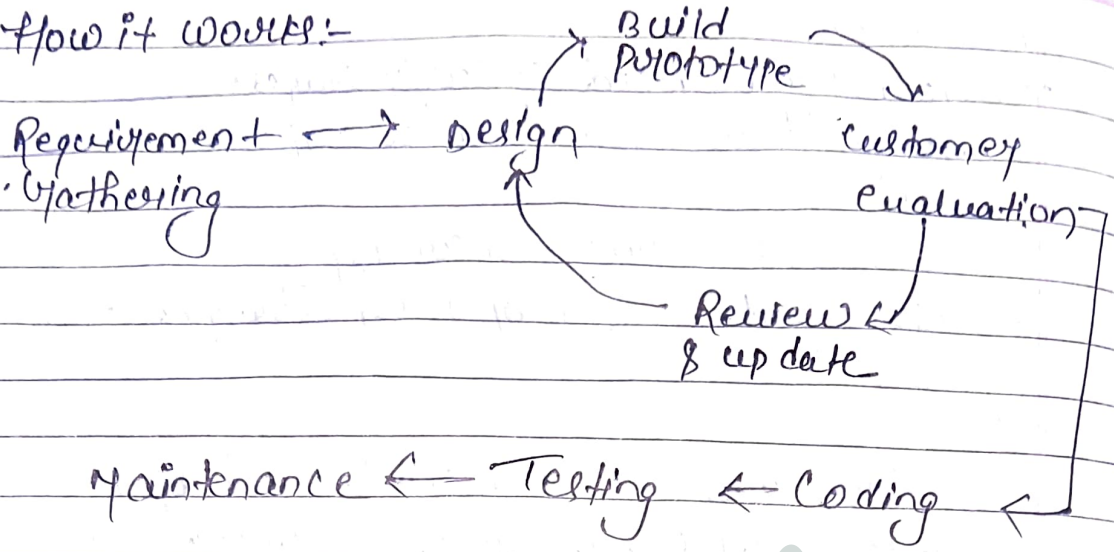
A model where a prototype (basic working version) is developed first to understand user needs before building the final sw.

Features :-
* Help when requirement are unclear.
- Involves users early in the process
- Can increase development time due to multiple revisions.

Advantage :-
- user give feedback early.
- Reduces chances of failure.

Disadvantage
- Time-consuming & expensive
- Frequent changes can delay final development.

How it works:-



(2) Agile process model.

* RAD (Rapid Application Development)
A fast-paced S/w development approach that focused on quick prototype and reusable components.

features
- Speed-focused, reduces development time
- use Component-based development.

Advantage
- Faster development & delivery
- User feedback ensure better final product.

Disadvantage
- Not suitable for large projects.
- Needs experienced developers.

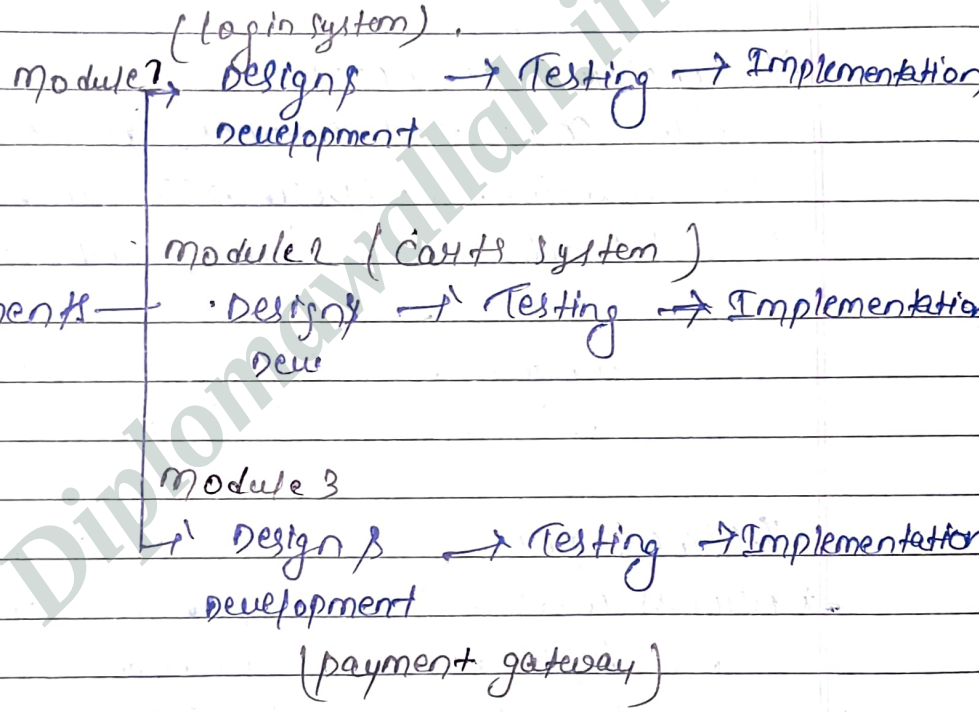
Requirement
↓
Quick prototype.

* Incremental model

The incremental model is a software development approach where the system is designed, implemented and tested in small parts (increments).

Feature

- develops software in stages (module by module)
- Each module is tested & improved separately.
- final product takes more time.



* Defined process (well-defined process)
→ A defined process is a process where every step is clearly understood, repeatable and predictable.
The input, output, and all stages are predefined and don't change much over time.
Ex- When a factory production its bottle. all bottle are in same size.

* Empirical process
→ An empirical process is based on observation, experience and feedback.
→ It used when things are not predictable.

Ex- Imagine we building a new mobile app and the client need keep changing.

Defined process	Empirical process
→ process where steps are defined.	process which is adapted based on experience, & feedback
→ predictable & controlled	Flexible and adaptive.
→ Requirement are well understood and stable.	Requirement are unclear evolving or likely to change
→ waterfall model, V-model.	Agile, Scrum, XP
→ project with fixed scope, cost and timeline	projects with uncertain or dynamic req.

large project → Chunks → Release → Feedback → Enhance

* Agile process:—

Agile is s/w dev. approach that focuses on flexibility, customer collaboration, and quick delivery of small functional parts of the s/w.

1. Agile Manifesto

The Agile manifesto was created in 2001 by a group of s/w developers to describe the core values and beliefs of Agile development.

* Four core values:— (core principle of Agile).

1. Individual and interactions over processes and tools.
2. Working Software and comprehensive documentations.
3. Customer collaboration over Contract negotiation.
4. Responding to change over following a plan.

2. Agile principles:—

There are 12 principles behind the Agile Manifesto.

- Customer satisfaction through early and continuous delivery.
- Welcome changing req., even late in develp.
- Deliver working s/w frequently.
- Close, daily cooperation b/w business people & developers.
- Build project around motivated individual.

- face to face conversation is the best way to communicate.
- working s/w is the primary measure of progress.
- Simplicity.

3. Agile practices :-
Agile use many practical techniques and frameworks such as :-

- Scrum - organizes work into small cycles called Scrums.
- Kanban - focuses on visualising work and limiting ongoing tasks.
- Daily Stand-ups - Short daily meeting to check progress.
- User Stories - Small requirements written in simple lang.

4. paradigm shift from plan-driven to Agile.
- plan-driven :- (old thinking)
traditional methods like waterfall focus on step by step plan :
(Requirement → design → Code → Test → deploy)
- These methods are rigid. If something changes it's hard to go back.

plan driven (old thinking)	Agile (new Thinking)
→ Heavy documentation	Working Software.
→ Fixed plan	Flexible and adative.
→ Client involved at the end.	Client involved throughout.
→ Sequential development	Iterative and incremental.
→ Change is risky	Change is welcome.

Sangem