# Module 1

04 August 2024 01:31

Software

- System Softwares
  - OS
  - Device Drivers (programs that allow OS to communicate with hardware)
  - Utility Programs (tools that perform maintenance tasks like Antivirus, Disk cleanup Backup, etc.)
- Application Softwares
  - Productivity Software
  - Web Browsers
  - Media Players
  - Graphics Software
  - Games
- Development Software
  - Integrated Development Environments (IDEs)
  - Compilers & Interpreters

## ✯ Concepts of Software Systems -

① Software Architecture high level structure; how components interact with each other

② Components & Modules individual units; specific functions; reusable; can combine

② Components & Modules : individual units ; specific functions ; reusable ; can combine

③ Interfaces : interaction btw Softwares & API's (Application Programming Interfaces)

④ Data Storage databases, file systems

⑤ Process & Threads independent execution units, threads ; smaller within a process

⑥ Middleware Software common services to applications outside OS ; web/app. servers

⑦ User Interface (UI) GUI's, CLI's, web interfaces

⑧ Networking & Communication Protocols like HTTP, TCP/IP

## ☆ Characteristics of Software Systems –

① Complexity

② Modularity breakdown into small, manageable parts to develop, test, maintain

③ Scalability handle increasing loads

④ Portability run on diff. hardware platforms

⑤ Reliability

⑥ Maintainability

⑦ Usability easy to use

⑧ Performance efficiency

⑨ Security unauthorised access, data breaches, other vulnerabilities

⑩ Interoperability work with other systems

⑪ Documentation comprehensive detailed info like design docs, user manuals, API's

## ☆ SDLC

The Software Development Life Cycle (SDLC) is a detailed process that helps development teams efficiently build the highest quality software at the lowest cost.

① Stage 1 Planning & Requirement Analysis

- Planning
- Define Project Scope
- Set objectives & Goals
- Resource Planning

② Stage 2 Defining Requirements

- Defining
- Functional Requirement
- Technical Requirement
- Requirement reviews & Approved

③ Stage 3 Design

- Design
- LLD (Low Level Design)
- HLD (High Level Design)

④ Stage 4 Development / Implementation

- Development
- Coding Standard
- Scalable Code
- Version Control
- Code Review

⑤ Stage 5 Testing & Integration

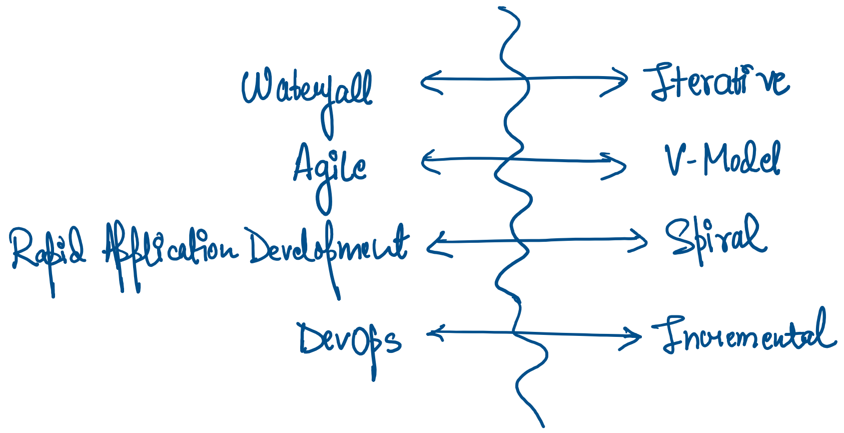
- System Testing
- Manual Testing
- Automated Testing

⑥ Maintenance & Support

- Deployment
- Release Planning
- Deployment Automation

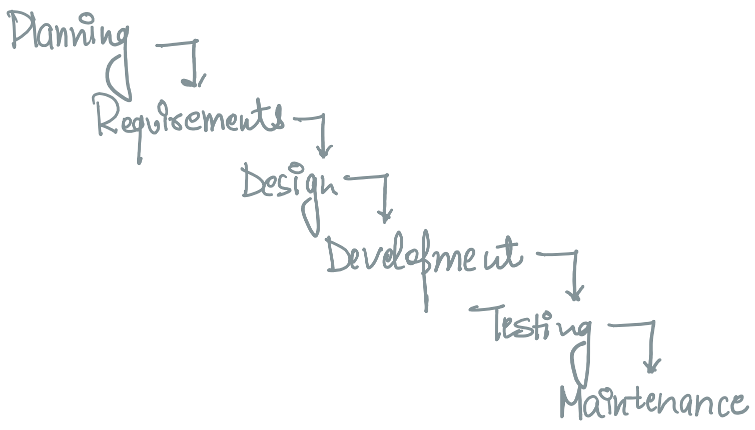
- Release Planning
- Deployment Automation
- Feedback

## ✱ Different Approaches / Models of Development



### ① WATERFALL MODEL

- Oldest & most straightforward
- Linear & Sequential Approach
- Each phase must be completed before moving onto the next



- Advantages : Simplicity
  Clear Documentation (each phase has its own set of documentation)
  Stable Requirements
  Predictability (in terms of timelines & deliverables)

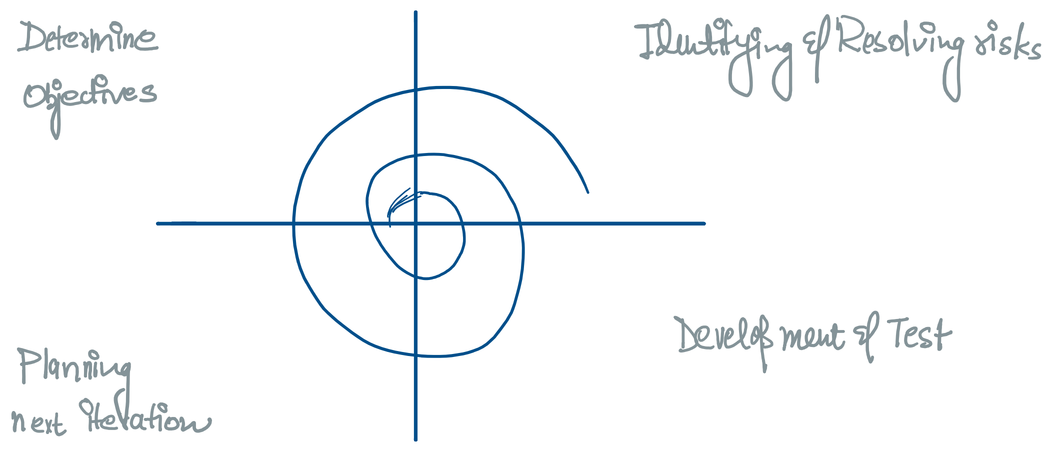
- Disadvantages : Rigidity
  Late Testing (testing after implementation)

- Disadvantages : Rigidity
  Late Testing (testing after implementation)
  Limited Client Involvement (only in initial phase)
  No prototyping

- When to Use?
  - Well Defined Requirements
  - Small to Medium Projects
  - Mission Critical Systems

## ② SPIRAL MODEL

- Iterative Development + Waterfall Model
- Loop represent phase
- Risks continously assessed & addressed



- Advantages : Risk Mitigation
  Flexibility in Requirements
  High Quality Products
  Client Involvement

- Disadvantages : Risk of not meeting Schedule or Budget
  Better for large projects
  Needs to be followed strictly
  More Documentation
  Not advisable for smaller projects

More Documentation

Not advisable for smaller projects

## ③ PROTOTYPE MODEL

- used when customers donot know exact project requirements
- most popular
- prototype ; test ; refine

$$\begin{tikzcd}
\text{Requirements} \arrow[r] & \text{Design} \arrow[r] & \text{Develop Prototype} \arrow[r] & \text{User Evaluation} \arrow[r] \arrow[r, bend right] & \text{Refining Prototype} \arrow[r] \arrow[l, bend right] & \text{Implement \& Maintain}
\end{tikzcd}$$

- Advantages : Partial product
  Customer Satisfaction
  Easily accomodates
  Missing functionalities easily figured out
  errors detected earlier
  Flexibility in Design
- Disadvantages : Cost concerning
  Uncertainity no. of Iterations befor prototyple finally accepted by Customer
- When to use?
  - Requirements not clearly understood
  - Requirements changing quickly