III B.Tech I Sem CSE

23CS503 : DevOps

Unit-II

Software development models and DevOps

Mr. P. Thirupathi
Assistant Professor, Department of CSE NRCM

Software Development Model

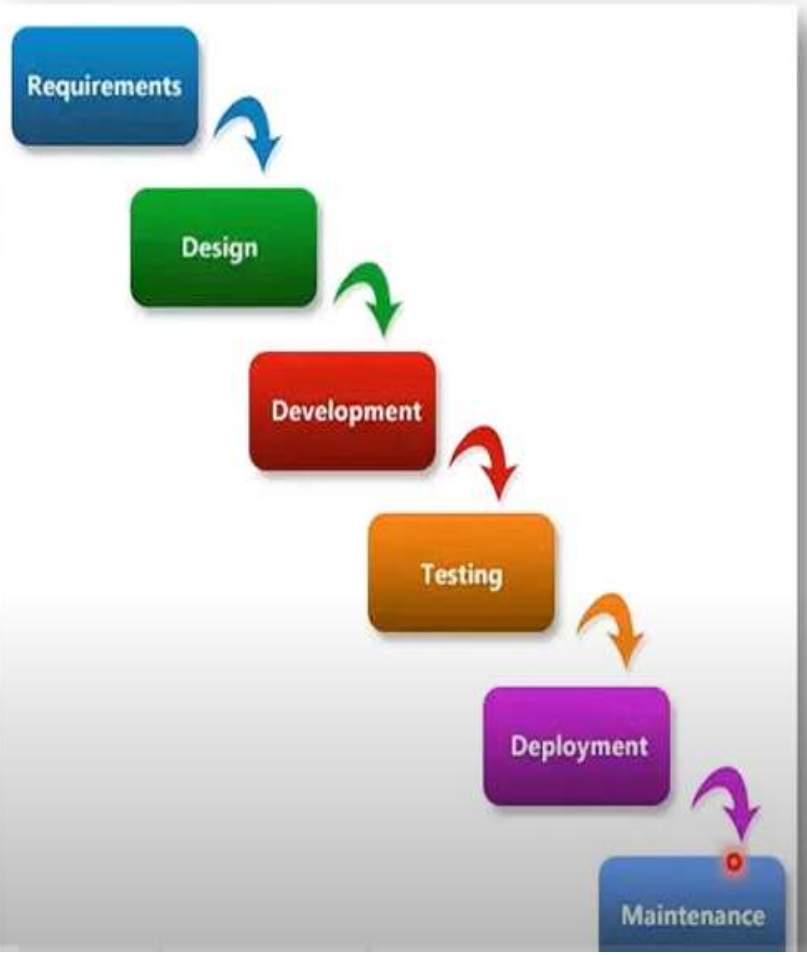
Software Development Models (also called **Software Development Life Cycle (SDLC) models**) are structured approaches or methodologies used to **plan, design, develop, test, and maintain** software systems in a systematic way.

Types of Models

1. **Waterfall Model** – Linear, sequential stages (Requirement → Design → Coding → Testing → Maintenance).
2. **V-Model** – Extension of Waterfall, each stage has a testing phase.
3. **Iterative Model** – Software built in repeated cycles (iterations).
4. **Spiral Model** – Combines iterative and waterfall with risk analysis.
5. **Agile Model** – Incremental, adaptive, customer collaboration focused.
6. **RAD Model** – Fast development using prototypes and reusable components.

➢ These models help in selecting the right development strategy based on project size, risk, and flexibility.

Waterfall Model



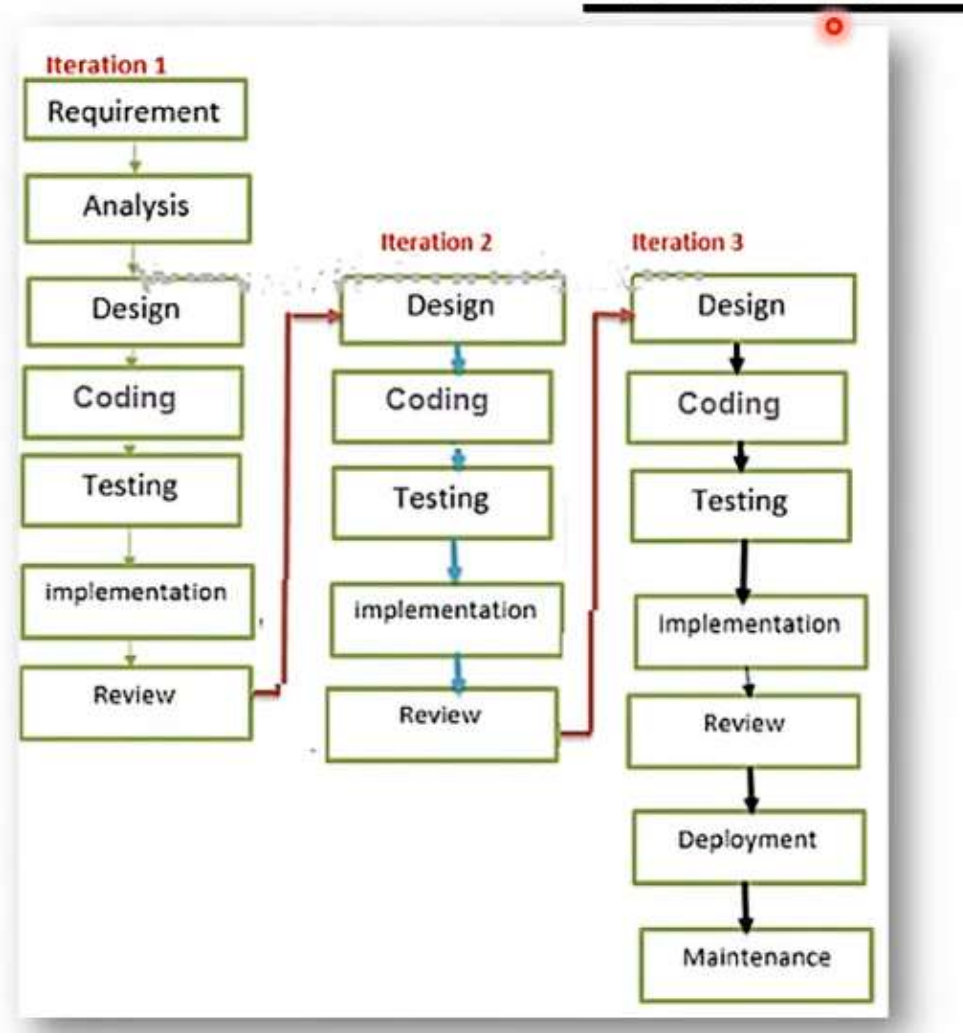
➢ **Features:**

- Fixed Requirements.
- Used in Small Projects.
- Cost Low.
- Good Documentation.

➢ **Disadvantages:**

- High risk.
- Low customer involvement.
- Errors difficult to recognized.

Iterative Model



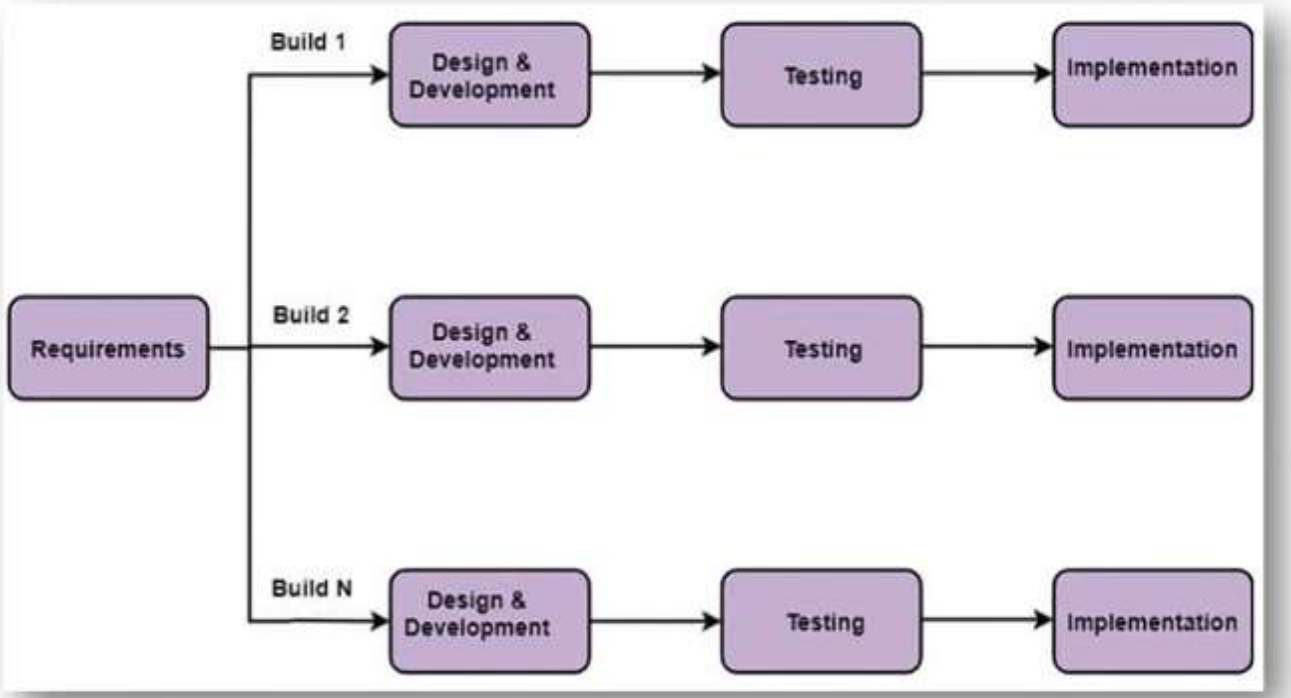
➢ **Features:**

- Changing Requirements.
- Used in Large Projects.
- Customer feedback taken.

➢ **Disadvantages:**

- More recourses required.
- Deadline not confirmed.
- Over budget.

Incremental Process Model



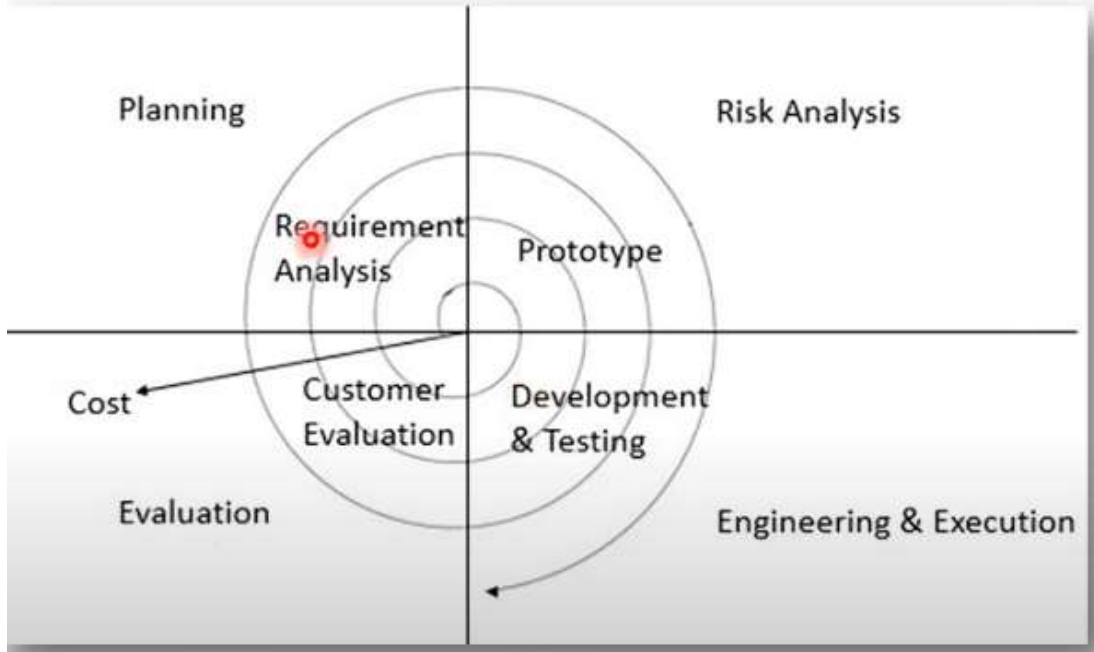
➢ **Features:**

- Fixed Requirements.
- Used in Large Projects.
- Quick release.
- Errors easy to recognized.

➢ **Disadvantages:**

- More recourses required.
- Good team & proper planning required.

Spiral Model



➢ **Features:**

- Used in Large & critical Projects.
- Risk identify & Analysis. Low risk.
- Used Prototyping method.
- Changing requirements.
- Customer feedback taken.
- Errors easy to recognized.

➢ **Disadvantages:**

- Requirements are unclear & complex.
- High budget.
- Excess documentation.

DevOps Architecture

DevOps **Architecture** refers to the structure, practices, and components that enable continuous integration (CI), continuous delivery (CD), automation, monitoring, and collaboration between development and operations teams. It combines tools, processes, and culture to deliver software efficiently and reliably.

Components of DevOps Architecture

- **Continuous Development**
  - Planning, coding, and version control.
  - Tools: Git, GitHub, GitLab, Bitbucket.
- **Continuous Integration (CI)**
  - Merging code frequently, automated builds, and testing.
  - Tools: Jenkins, Travis CI, CircleCI, GitHub Actions.
- **Continuous Testing**
  - Automated testing to identify bugs early.
  - Tools: Selenium, JUnit, TestNG.
- **Continuous Deployment/Delivery (CD)**
  - Deploying code automatically to different environments.
  - Tools: Jenkins, Spinnaker, ArgoCD.

- **Continuous Monitoring**
  - Real-time monitoring of applications and infrastructure.
  - Tools: Prometheus, Nagios, Grafana, ELK Stack.
- **Configuration Management**
  - Managing infrastructure with code.
  - Tools: Ansible, Puppet, Chef.
- **Containerization & Orchestration**
  - Deploying applications consistently across environments.
  - Tools: Docker, Kubernetes.
- **Cloud Infrastructure**
  - Hosting applications and scaling them dynamically.
  - Providers: AWS, Azure, Google Cloud.

DevOps Architecture Workflow

1. **Code** → Developers write code and push to version control.
2. **Build** → CI server compiles and builds the application.
3. **Test** → Automated tests run to check quality.
4. **Release** → Code is prepared for deployment.
5. **Deploy** → Application is deployed to staging/production.
6. **Operate** → Infrastructure automation ensures smooth operations.
7. **Monitor** → System and application are monitored for performance issues.
8. **Feedback** → Insights are fed back to improve development.

DevOps Architecture Workflow

[Plan]

↓

[Code] --(GitHub/GitLab/Bitbucket)

↓

[Build & CI] --(Jenkins/CircleCI)

↓

[Test] --(Selenium/JUnit)

↓

[Release/Deploy] --(Docker/Kubernetes, Ansible)

↓

[Operate] --(Cloud Infrastructure: AWS/Azure/GCP)

↓

[Monitor] --(Prometheus/Grafana/ELK)

↓

[Feedback → Back to Plan]

DevOps architecture is a continuous loop of **planning** → **coding** → **building** → **testing** → **releasing** → **deploying** → **operating** → **monitoring** → **feedback**, supported by automation and tools.