

Introduction to Software Development

Introduction to Software Development

Key concepts, phases, roles, and best practices

1. What Is Software Development?

Aspect	Description
Definition	The systematic process of designing, creating, testing, and maintaining software applications.
Goal	Transform user needs or business requirements into functional, reliable, and maintainable software.
Scope	Ranges from a single script to large, multi-team enterprise systems.
Disciplines Involved	Requirements engineering, architecture, coding, testing, deployment, maintenance, project management, and quality assurance.

2. Core Development Life Cycle Models

Model	When to Use	Main Phases	Key Characteristics
Waterfall	Projects with stable, well-understood requirements.	Requirements ? Design ? Implementation ? Verification ? Maintenance	Linear, documentation-heavy, easy to manage milestones.
Iterative / Incremental	Need early delivery of partial functionality.	Repeat cycles of planning, design, coding, testing.	Each iteration adds value; feedback incorporated continuously.
Agile (Scrum, Kanban, XP)	Dynamic environments, frequent change, cross-functional teams.	Sprint planning ? Development ? Review ? Retrospective (Scrum)	Emphasizes collaboration, rapid delivery, continuous improvement.
DevOps / Continuous Delivery	High-frequency releases, automation focus.	Code ? Build ? Test ? Release ? Operate ? Monitor	Tight integration of development and operations; pipelines automate repeatable steps.

Introduction to Software Development

Model	When to Use	Main Phases	Key Characteristics
V?Model	Safety?critical or regulated domains.	Parallel development & testing phases (unit, integration, system, acceptance).	Emphasizes verification & validation at each stage.

3. Major Phases of a Typical Project

1. Requirement Gathering & Analysis

- Stakeholder interviews, user stories, use cases.
- Produce **SRS** (Software Requirements Specification).

2. System Design

- **High?level (architectural)**: component diagram, technology stack, data flow.
- **Low?level (detailed)**: class diagrams, API contracts, database schema.

3. Implementation (Coding)

- Follow coding standards, use version control (Git).
- Apply design patterns where appropriate.

4. Testing

- **Unit, Integration, System, Acceptance, Performance, Security** tests.
- Automated test suites (JUnit, pytest, Selenium, etc.).

5. Deployment

- Build artifacts, containerization (Docker), orchestration (Kubernetes), cloud services.
- Use CI/CD pipelines (Jenkins, GitHub Actions, GitLab CI).

6. Maintenance & Support

- Bug fixing, enhancements, technical debt reduction.
 - Monitoring (Prometheus, ELK) and incident response.
-

4. Key Roles & Responsibilities

Role	Primary Focus	Typical Deliverables
Product Owner / Business Analyst	Define <i>what</i> to build	Backlog, user stories, acceptance criteria
Software Architect	Define <i>how</i> to build (overall structure)	Architecture diagrams, technology decisions

Introduction to Software Development

Role	Primary Focus	Typical Deliverables
Developer / Engineer	Write <i>code</i>	Source code, unit tests, documentation
QA Engineer / Tester	Verify <i>quality</i>	Test plans, automated test scripts, defect reports
DevOps Engineer	Bridge <i>development</i> and <i>operations</i>	CI/CD pipelines, infrastructure as code, monitoring