

Introduction to Software Development

Introduction to Software Development

Key concepts, phases, and best practices for building reliable software systems.

1. What Is Software Development?

Aspect	Description
Definition	The systematic process of designing, creating, testing, and maintaining computer programs and applications.
Goal	Transform user needs or business requirements into functional, maintainable, and performant software.
Stakeholders	• End-users • Product owners / business analysts • Developers (front-end, back-end, full-stack) • QA engineers • DevOps / operations teams • Project managers
Typical Outputs	Executable applications, libraries, APIs, scripts, documentation, and deployment artefacts.

2. Core Concepts

Concept	Why It Matters	Typical Tools / Techniques
Requirements Engineering	Captures <i>what</i> the system must do.	User stories, use-case diagrams, requirement specs, JIRA, Confluence
Software Architecture	Defines high-level structure, technology stack, and interaction patterns.	Layered architecture, microservices, MVC, UML diagrams, C4 model
Algorithm & Data Structure Design	Determines <i>how</i> problems are solved efficiently.	Big-O analysis, sorting/searching algorithms, trees, graphs
Version Control	Tracks changes, enables collaboration, and supports rollback.	Git (GitHub, GitLab, Bitbucket), SVN

Introduction to Software Development

Concept	Why It Matters	Typical Tools / Techniques
Testing	Verifies correctness, performance, and security.	Unit, integration, system, UI, performance, security testing; frameworks (JUnit, pytest, Selenium)
Continuous Integration / Continuous Delivery (CI/CD)	Automates build, test, and deployment pipelines.	Jenkins, GitHub Actions, GitLab CI, CircleCI
DevOps & Infrastructure as Code	Bridges development and operations for reliable releases.	Docker, Kubernetes, Terraform, Ansible
Maintenance & Refactoring	Keeps software usable, secure, and adaptable over time.	Code reviews, static analysis, technical debt tracking

3. Development Lifecycle Models

Model	Main Idea	Strengths	Weaknesses
Waterfall	Sequential phases (requirements ? design ? implementation ? verification ? maintenance).	Simple, easy to manage documentation.	Inflexible to change; late testing.
Iterative	Repeats cycles of planning, development, and evaluation.	Early feedback, incremental value.	Requires good iteration planning.
Agile (Scrum, Kanban)	Short sprints/continuous flow, frequent stakeholder interaction.	High adaptability, fast delivery.	Needs disciplined team communication.
DevOps?Enabled Continuous Delivery	Combines Agile with automated pipelines for rapid, reliable releases.	Short lead time, high quality.	Demands mature automation & monitoring.
Hybrid (e.g., SAE, LeSS)	Scales Agile principles for large organisations.	Balances coordination and flexibility.	Can become bureaucratic if not tuned.

4. Typical Phases of a Software Project

Introduction to Software Development

1. Planning & Feasibility

- Define scope, budget, timeline, risk assessment.
- Deliverables: project charter, high-level roadmap.

2. Requirements Gathering

- Elicit functional & non-functional requirements.
- Techniques: interviews, workshops, surveys, prototyping.

3. Design

- **High-level (architecture):** component