

Introduction to Software Development

Introduction to Software Development

Key concepts, phases, roles, and best practices

1. What Is Software Development?

- **Definition** – The systematic process of designing, creating, testing, and maintaining software applications or systems.
 - **Goal** – Transform user needs and business requirements into functional, reliable, and maintainable software.
 - **Scope** – Ranges from tiny scripts and mobile apps to large-scale enterprise systems and cloud platforms.
-

2. Core Concepts

Concept	Description	Typical Tools / Techniques
Programming Languages	Formal languages used to write code (e.g., Python, Java, JavaScript, C#).	IDEs, compilers, interpreters
Algorithms & Data Structures	Foundations for solving problems efficiently.	Pseudocode, Big-O analysis
Version Control	Tracks changes, enables collaboration, and preserves history.	Git, Mercurial, Subversion
Software Architecture	High-level structure of a system (layers, modules, services).	UML, design patterns, micro-services
Testing	Verifies correctness, performance, and security.	Unit, integration, system, acceptance tests; frameworks (JUnit, pytest)
Deployment	Moving code from development to production environments.	CI/CD pipelines, Docker, Kubernetes, cloud services
Maintenance	Ongoing bug fixes, enhancements, and adaptation to new requirements.	Issue trackers, monitoring tools

Introduction to Software Development

3. Development Lifecycle Models

Model	When to Use	Key Characteristics
Waterfall	Small, well-defined projects with stable requirements.	Linear, sequential phases; heavy documentation.
Iterative / Incremental	Projects needing early feedback and gradual refinement.	Repeated cycles; each iteration adds functionality.
Agile (Scrum, Kanban)	Dynamic environments, frequent changes, cross-functional teams.	Short sprints, daily stand-ups, backlog grooming, continuous delivery.
DevOps	Organizations emphasizing collaboration between development and operations.	Automation, monitoring, rapid release cycles, culture of shared responsibility.
Spiral	High-risk, large-scale systems where risk analysis is critical.	Combines iterative development with risk assessment at each loop.

4. Typical Phases of a Software Project

1. Requirement Analysis

- Gather functional & non-functional requirements.
- Produce specifications (user stories, use cases, requirement docs).

2. Design

- Define system architecture, component interaction, and UI/UX.
- Create diagrams (UML, flowcharts) and prototypes.

3. Implementation (Coding)

- Write source code following coding standards.
- Perform peer reviews / pull-request checks.

4. Testing

- Execute automated and manual tests.
- Log defects, verify fixes, and assess performance/security.

Introduction to Software Development

5. Deployment

- Package the application (containers, installers).
- Deploy to staging, then production; configure rollback mechanisms.

6. Maintenance & Support

- Monitor production, address incidents, release patches.
- Gather user feedback for future enhancements.

5. Key Roles in a Development Team

Role	Primary Responsibilities
Product Owner / Business Analyst	Define vision, prioritize backlog, communicate stakeholder needs.
Project Manager / Scrum Master	Facilitate process, remove impediments, track progress.
Software Engineer / Developer	Design, code, unit?test, and ref