

Introduction to Software Development

Introduction to Software Development

Key concepts, phases, and best practices

1. What Is Software Development?

Aspect	Description
Definition	The systematic process of conceiving, designing, building, testing, and maintaining software applications.
Goal	Deliver functional, reliable, and maintainable software that solves a specific problem or fulfills a user need.
Stakeholders	Developers, designers, testers, product owners, end-users, and business managers.

2. Core Concepts

Concept	Explanation	Typical Tools / Techniques
Requirement Analysis	Identify what the software must do (functional) and constraints (non-functional).	Interviews, user stories, use case diagrams, JIRA, Confluence
Software Architecture	High-level structure that defines components, their responsibilities, and interactions.	Layered architecture, microservices, MVC, UML component diagrams
Programming Paradigms	Ways of structuring code (e.g., procedural, object-oriented, functional, reactive).	Language features, design patterns
Version Control	Track changes, enable collaboration, and manage releases.	Git, Mercurial, Subversion
Testing	Verify that software works as intended and remains stable over time.	Unit, integration, system, acceptance testing; frameworks (JUnit, pytest, Selenium)

Introduction to Software Development

Concept	Explanation	Typical Tools / Techniques
Continuous Integration / Continuous Delivery (CI/CD)	Automate building, testing, and deployment to reduce manual errors.	Jenkins, GitHub Actions, GitLab CI, Azure Pipelines
Maintenance	Ongoing updates, bug fixes, performance tuning, and adaptation to new environments.	Issue trackers, monitoring tools (Prometheus, New Relic)

3. Development Life Cycle Models

Model	When to Use	Key Characteristics
Waterfall	Small, well-defined projects with stable requirements.	Sequential phases, no backtracking.
Iterative	Projects needing early feedback and gradual refinement.	Repeats cycles of planning, development, and evaluation.
Agile (Scrum, Kanban)	Dynamic environments with frequent requirement changes.	Short sprints, daily standups, backlog grooming, continuous stakeholder involvement.
DevOps	Organizations emphasizing rapid delivery and operational stability.	Merges development and operations, heavy automation, culture of shared responsibility.

4. Typical Workflow (Agile-oriented)

1. **Backlog Refinement** – Prioritize user stories, estimate effort (story points, t-shirts).
 2. **Sprint Planning** – Select items for the upcoming sprint, define the sprint goal.
 3. **Development** – Write code, commit to version control, peer-review (pull requests).
 4. **Continuous Integration** – Automated builds & unit tests run on each commit.
 5. **Testing** – QA executes automated and manual tests; bugs are logged.
 6. **Demo / Review** – Team shows completed work to stakeholders; gathers feedback.
 7. **Retrospective** – Reflect on process, identify improvements for the next sprint.
-

5. Essential Skills for Developers

Introduction to Software Development

Category

Specific Skills

Technical Proficiency in at least one programming language, understanding of data structures & algorithms, familiarity with databases (SQL/NoSQL), API design, debugging.

Tooling Git, IDEs (VS Code, IntelliJ), containerization (Docker), cloud platforms (