tags: - cocubes - technical - sdlc - agile

Software Development Methodologies

SDLC phases

1. Requirements: determine what to build.
2. Analysis/planning: assess scope, feasibility, risks and approach.
3. Design: architecture, database, interfaces and detailed design.
4. Implementation: code.
5. Testing: verify/validate quality.
6. Deployment: release to users/production.
7. Maintenance: fix, enhance and operate.

The exact names may differ, but the flow is stable.

Method comparison

Model	Strength	Weakness	Best fit
Waterfall	clear sequential documentation	costly late change	stable, fixed requirements
V-model	testing mapped to development phases	rigid to change	quality-critical stable requirements
Iterative/ incremental	deliver in pieces and learn	needs planning/control	evolving product
Spiral	risk analysis in cycles	complex and costly	high-risk large project
Prototype	clarify uncertain requirements	prototype may be mistaken for final product	unclear UI/requirements
Agile	short iterations, feedback, adaptation	requires active collaboration	changing requirements
DevOps	development + operations, automation/continuous delivery	cultural/tooling investment	frequent reliable releases

Agile vocabulary

- Product backlog: prioritised list of work.
- Sprint: time-boxed iteration.
- User story: small user-focused requirement.
- Sprint planning: choose/plan sprint work.
- Daily scrum: short coordination meeting.

- Review: inspect the increment with stakeholders.
- Retrospective: improve the team's process.
- Scrum master: facilitates Scrum/removes impediments.
- Product owner: owns priorities/value.

CI, CD and version control

- Continuous integration: frequently merge code and automatically build/test.
- Continuous delivery: software is always releasable; deployment may be approved manually.
- Continuous deployment: successful changes automatically reach production.
- Git supports version control, branches, merging and history.

Requirement concepts

- Functional requirement: what the system does, for example "user can reset password."
- Non-functional requirement: quality constraint, for example response time, availability, security or usability.

Verification versus validation

- Verification: "Are we building the product right?" Checks against specifications, reviews and static analysis.
- Validation: "Are we building the right product?" Checks that the product meets user needs, often via execution/testing.

[[03_Technical/Software_QA_and_Testing|QA and testing]] · [[03_Technical/Technical_Quick_Recall|Quick recall]]