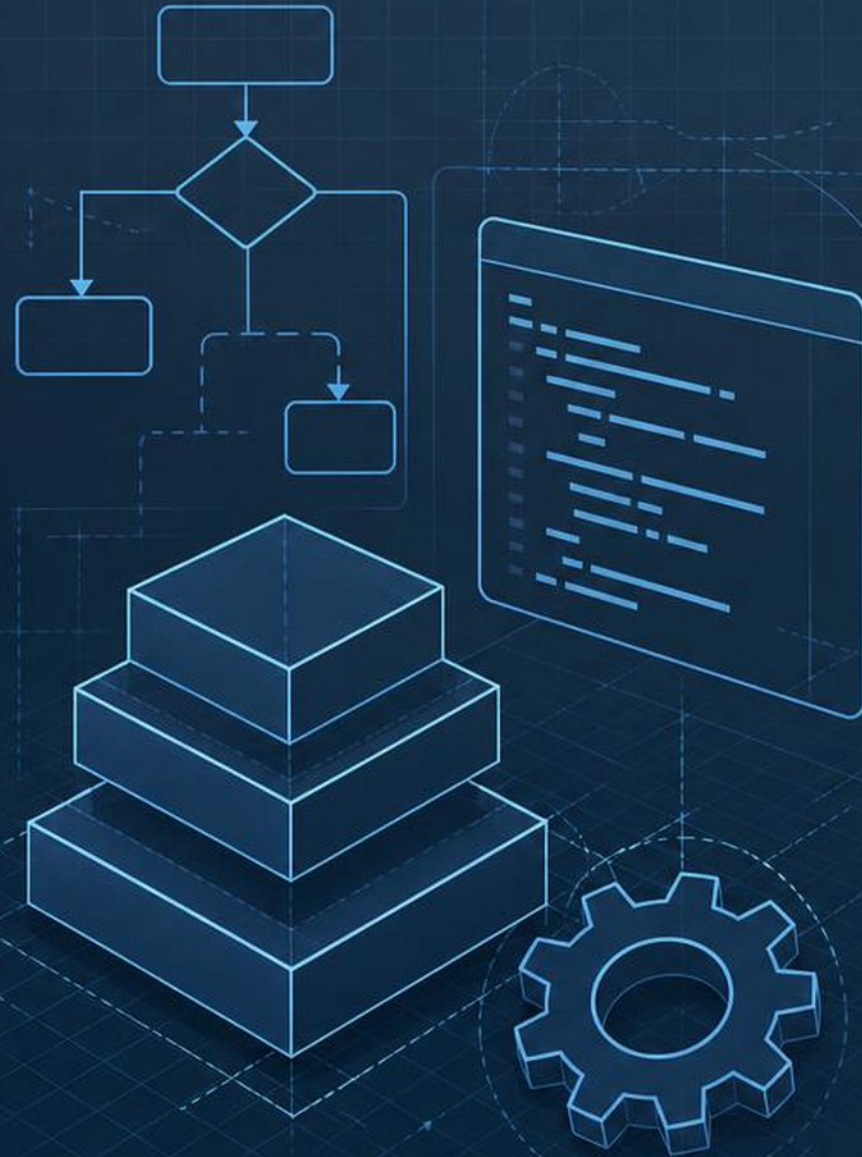


Software Engineering Fundamentals

- Systematic, disciplined, quantifiable approach to building and maintaining software
- More than coding: includes design, testing, deployment, and teamwork
- Applies engineering principles to manage complexity and ensure quality
- In 2026, AI shifts focus to verification, architecture, and engineering judgment



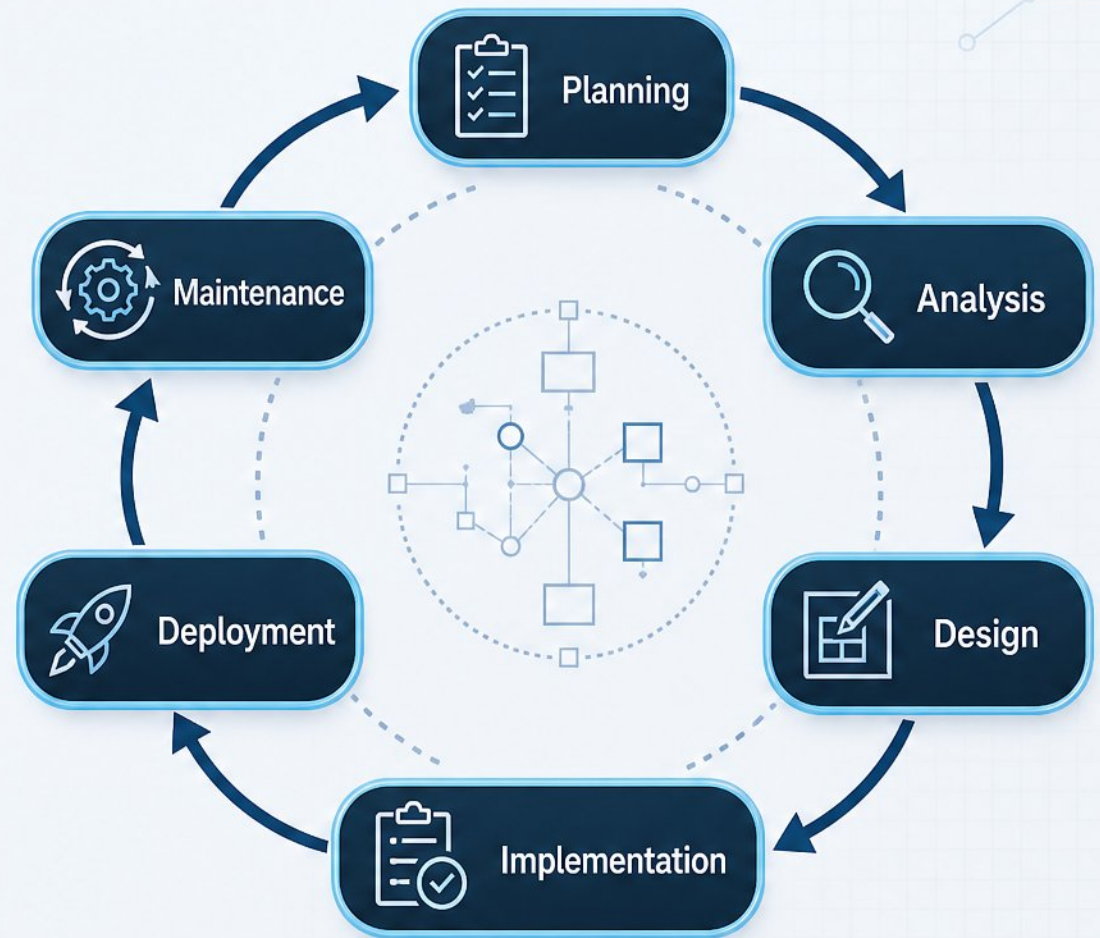
The Modern Engineering Mindset

- - From code writer to system thinker
- - AI assistants shift focus to verification and judgment
- - Orchestration and discernment define elite engineers
- - Past failures reinforce the need for rigor



The Software Development Life Cycle

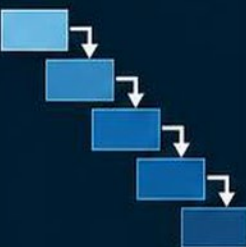
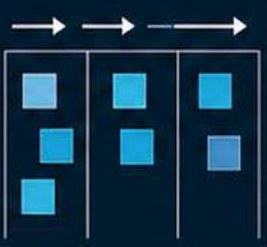
- - Phases: planning, analysis, design, implementation, testing, deployment, maintenance
- - A structured lifecycle reduces rework, delays, and budget overruns
- - Modern SDLCs integrate security and continuous improvement
- - Trade-offs always exist between speed, quality, and scope



Choosing an SDLC Model

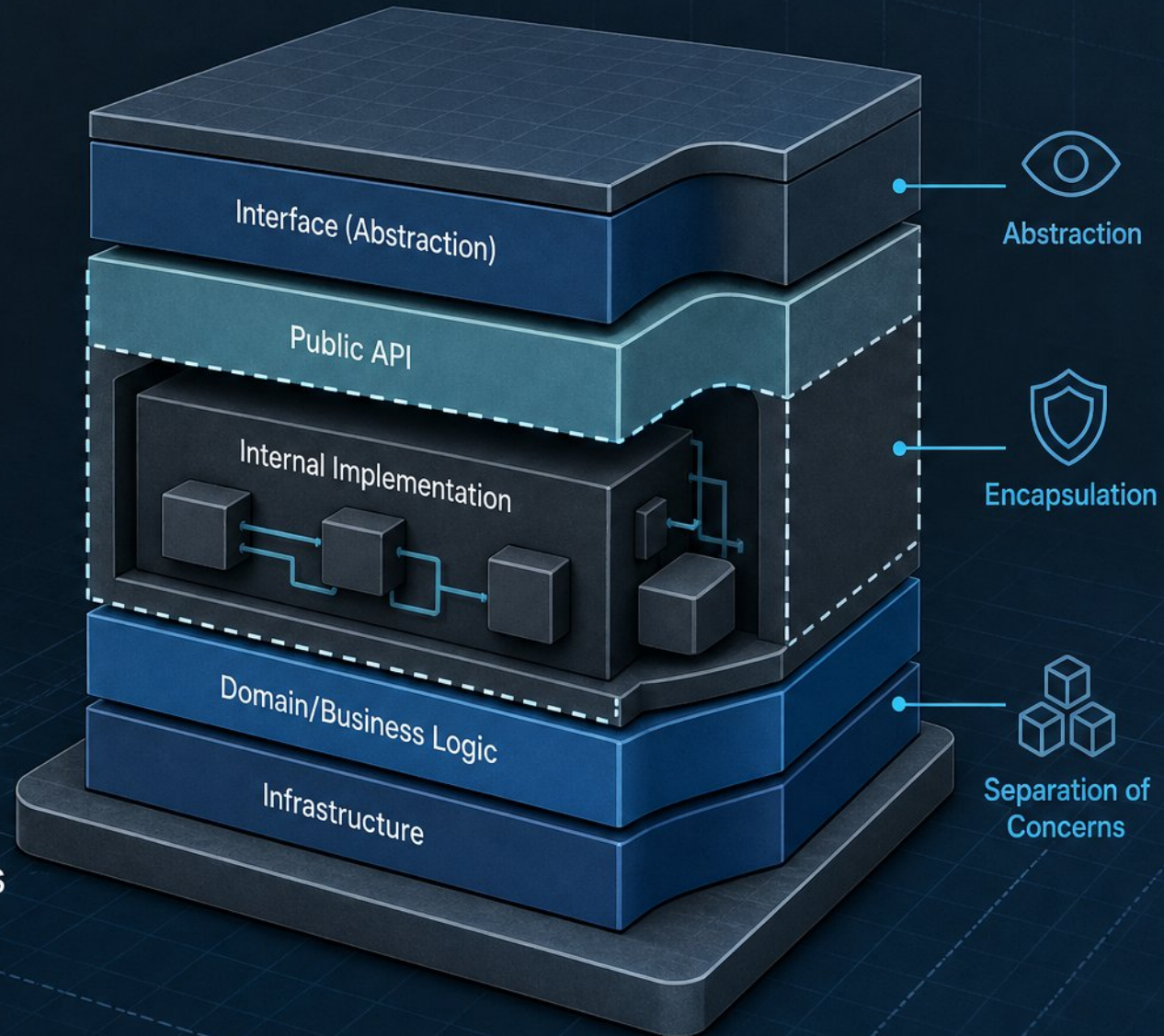
- Waterfall, Agile, Scrum, Kanban, Spiral, V-Model, and DevOps compared
- Decision framework: requirements stability, release cadence, team maturity
- Hybrid models dominate enterprise delivery in 2026
- Practical examples of when each model fits



Waterfall	Agile	Scrum	Kanban	Spiral	V-Model	DevOps
						
Linear, sequential phases flow downward. Best for stable requirements and well-defined scope.	Iterative and incremental development with continuous feedback and adaptation.	Agile framework with time-boxed sprints, roles, ceremonies, and backlog.	Visual workflow management with work-in-progress limits and continuous flow.	Risk-driven iterative cycles with planning, analysis, prototyping, and evaluation.	Verification and validation mapped to each development phase.	End-to-end automation and collaboration across development and operations.
Fits when requirements are clear and unlikely to change.	Fits when requirements evolve and frequent delivery is valuable.	Fits when teams can commit to regular sprints and clear prioritization.	Fits when work is ongoing and demand is variable or unpredictable.	Fits when projects are complex and risk management is critical.	Fits when quality, compliance, and traceability are essential.	Fits when rapid delivery, reliability, and continuous improvement are key.

Core Engineering Principles

- Separate concerns through modularity and abstraction
- Hide internal details with encapsulation
- Apply DRY, KISS, and YAGNI pragmatically
- Aim for high cohesion and low coupling
- Use high-level design patterns to structure systems





Requirements and Project Planning



- Distinguish functional vs. non-functional requirements early



- Capture work as user stories with clear acceptance criteria



- Estimate with story points, planning poker, or t-shirt sizing



- Prioritize ruthlessly using the MoSCoW framework



- Maintain a healthy backlog that reflects current priorities

USER STORY

As a user,
I want to reset my password,
so that I can regain access
to my account.

ACCEPTANCE CRITERIA

- ☒ User receives an email with a secure reset link
- ☒ User can set a new password that meets the policy
- ☒ User is able to sign in with the new password

T-SHIRT SIZING



Must
Have

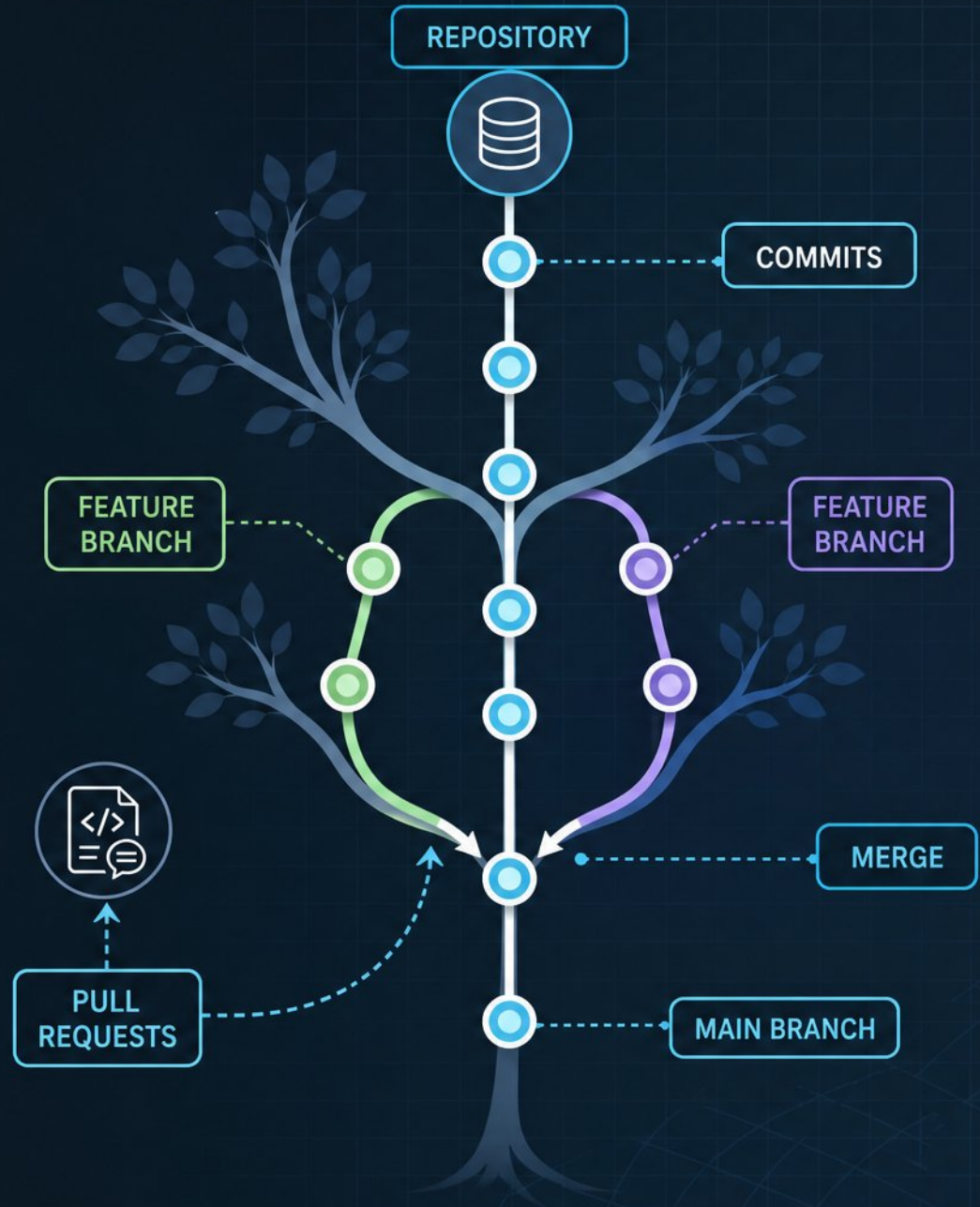
Should
Have

Could
Have

Won't
Have

Version Control and Collaboration

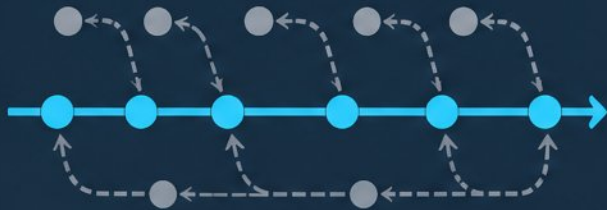
- Version control tracks changes and synchronizes team contributions
- Git repositories, commits, branches, and merges form the core workflow
- Feature branches and pull requests isolate work for structured code review
- Merge frequently to avoid conflicts and keep main deployable



Branching Strategies That Scale



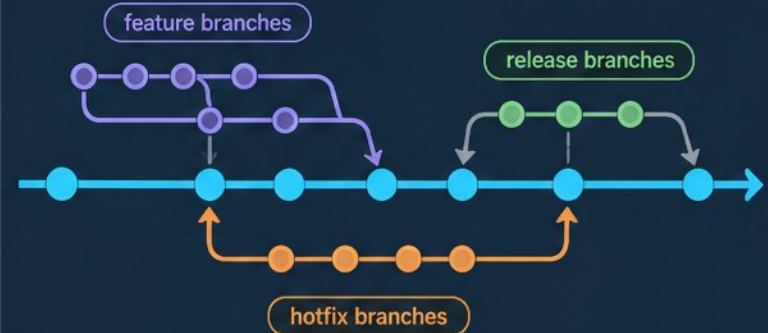
TRUNK-BASED DEVELOPMENT



GITHUB FLOW



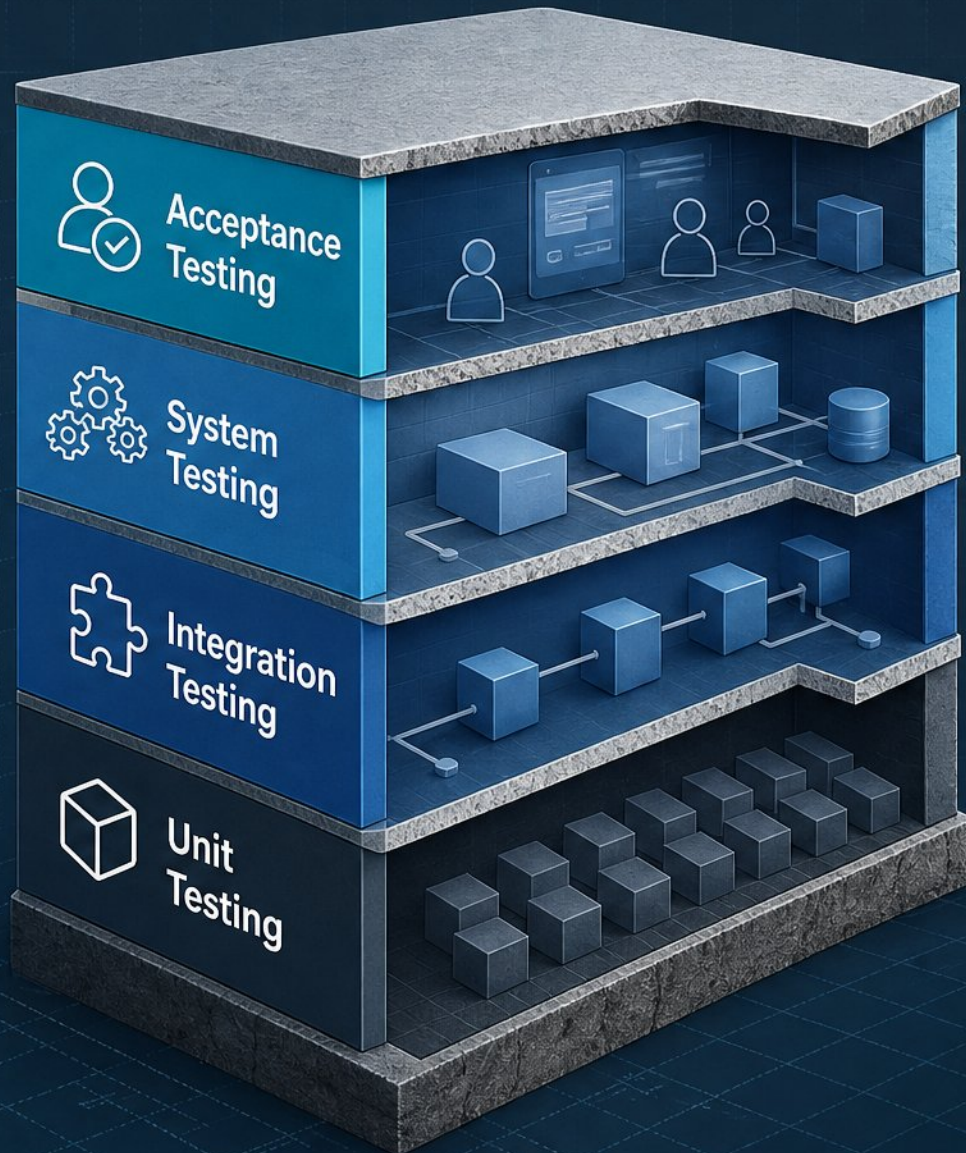
GITFLOW



- - Trunk-based development, GitHub Flow, GitFlow, and GitLab Flow compared
- - Branching impacts CI/CD speed, release cadence, and hotfix response
- - DORA research links short branch lifetimes to elite team performance
- - Feature flags enable gradual rollout without long-lived branches
- - Choose a model that matches team maturity and deployment rhythm



Testing and Quality Assurance Fundamentals



- Testing is a continuous discipline, integrated into every commit



- Unit, integration, system, and acceptance testing form the layers



- Prefer automation: fast tests first, reserve manual for exploration



- TDD guides design and drives code through failing tests



- Continuous testing in CI/CD catches defects when they are cheapest to fix

A Practical Testing Strategy



- Test pyramid, testing trophy, and diamond compared



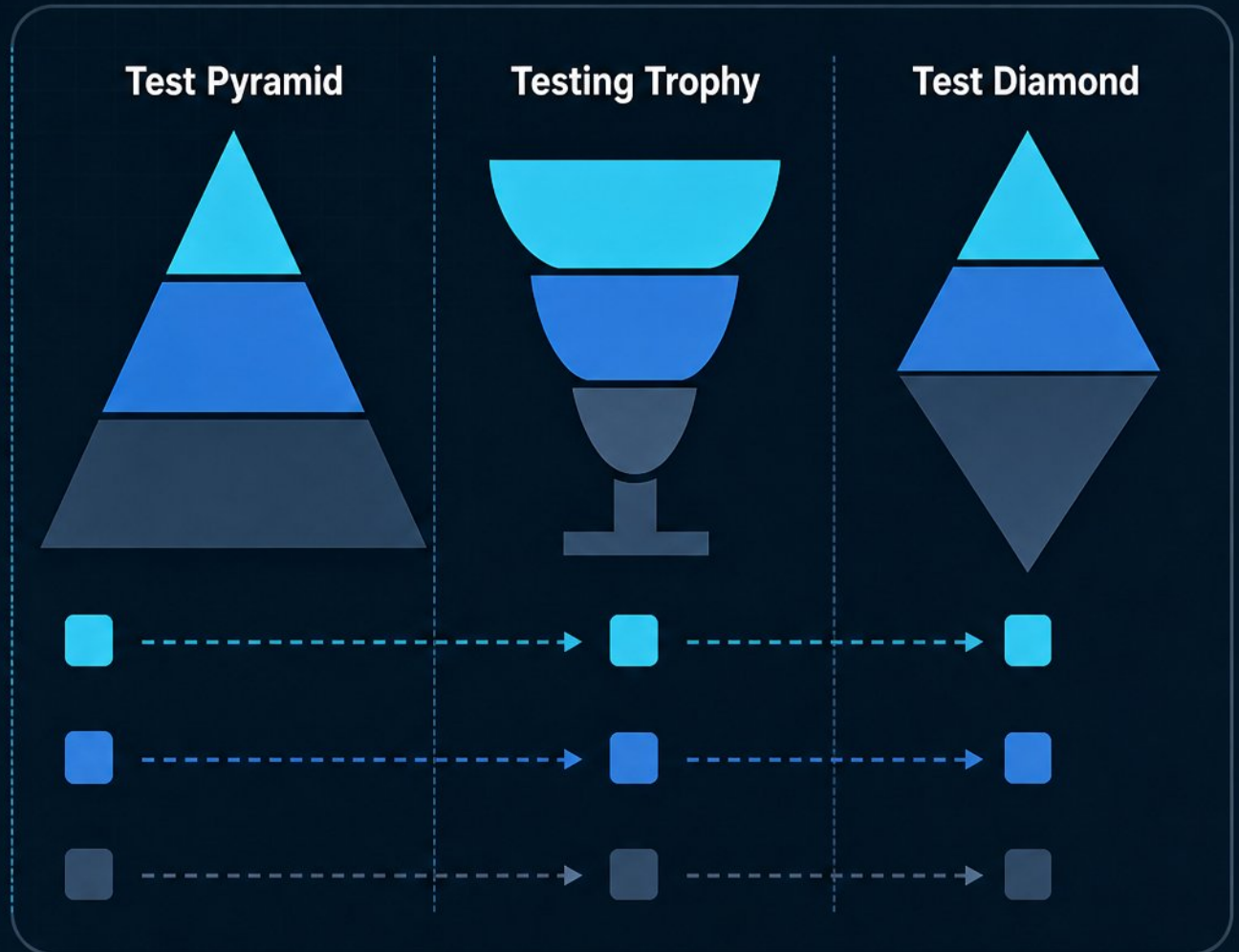
- Google's Small/Medium/Large size model as a measurable framework



- Coverage targets chosen by risk, not by mandate



- AI-generated tests and the rise of mutation testing

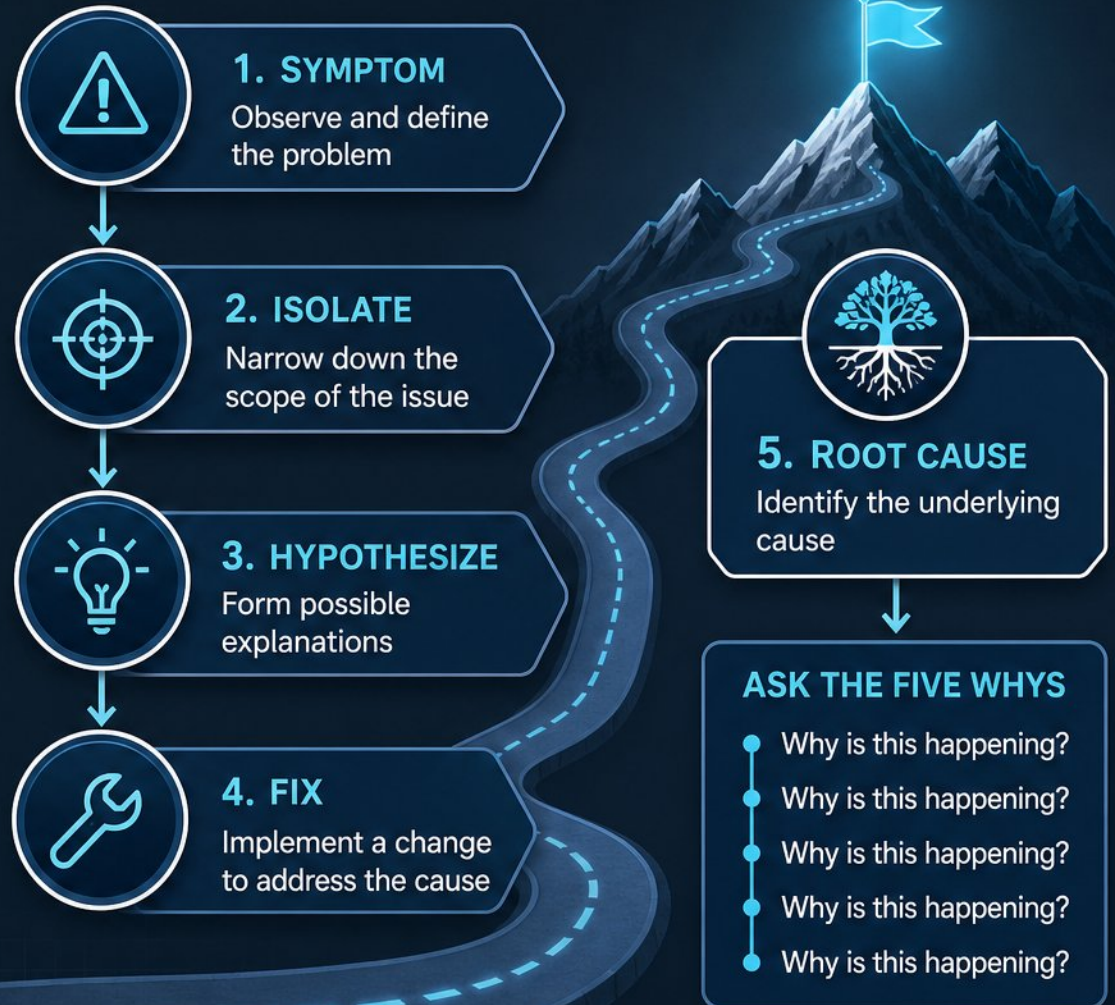


[illegible]

- ✓ - Readable, self-documenting code with clear naming conventions
- ✓ - Recognizing code smells that signal deeper structural problems
- ✓ - Safe refactoring techniques to improve existing code
- ✓ - Managing technical debt for long-term project health

Debugging and Problem Solving

- Apply systematic debugging: reproduce, isolate, hypothesize, fix
- Read error messages, parse logs, use interactive debuggers
- Perform root cause analysis; ask the Five Whys
- Debug AI-influenced code with verification and discernment



A systematic journey from symptom to root cause.

Tools, Environments, and DevOps Basics

- Development environments, dependency management, and Docker containers
- CI/CD automates build, test, and deployment for safe releases
- Popular tools: GitHub Actions, GitLab CI, Terraform, Kubernetes
- Observability means metrics, logs, and traces for reliable systems
- Governance now uses policy-as-code and a single audit trail



Career Pathways and Next Steps

- Common roles: frontend, backend, full-stack, data, AI, and DevOps engineers
- Specialize at senior levels; generalists win early career roles
- Show finished projects, open-source work, and hands-on experience
- Keep GitHub current; employers screen shipped work over credentials
- Learn AI tools deeply—they are now baseline expectations for engineers



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/ce6d3cd0/public