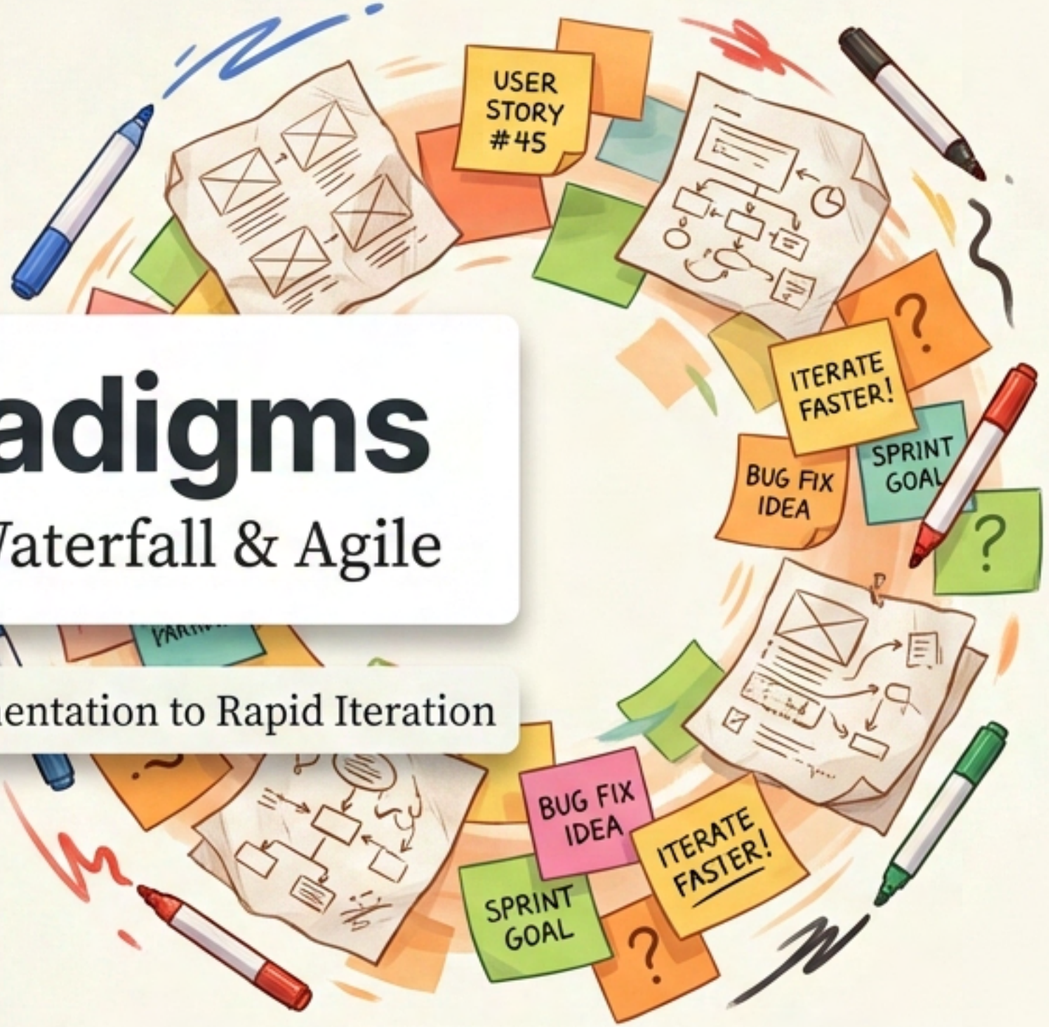
Structure.

Speed.



Testing Paradigms

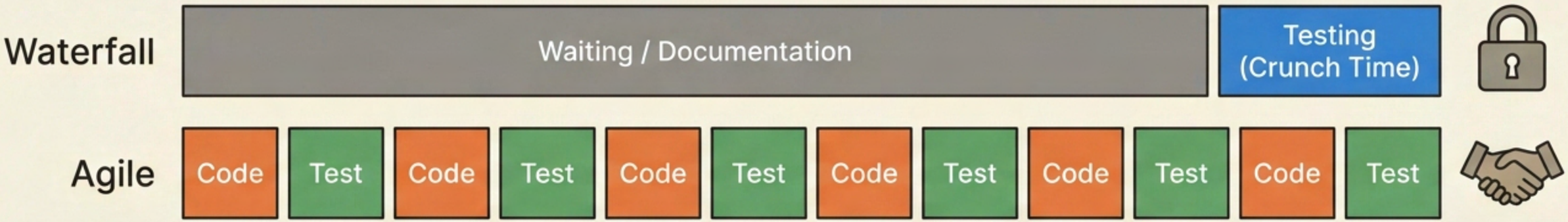
The Black Box Guide to Waterfall & Agile

Navigating the SDLC: From Strict Documentation to Rapid Iteration

A Learning Module by Skill-Wanderer

Why the Development Model Defines Your Day

While Black Box testing always focuses on inputs and outputs, the Software Development Lifecycle (SDLC) dictates your timing, your artifacts, and your team dynamics. It shifts your role from Gatekeeper to Collaborator.



Timing

Waterfall: Long wait, then rushed.

Agile: Continuous, daily testing.

Documentation

Waterfall: Detailed Proof (Step-by-step cases).

Agile: Speed & Intuition (Checklists/Exploratory).

The Role

Waterfall: The Gatekeeper.

Agile: The Collaborator.

Key Takeaway: In Waterfall, you verify against a document. In Agile, you validate value for a user.

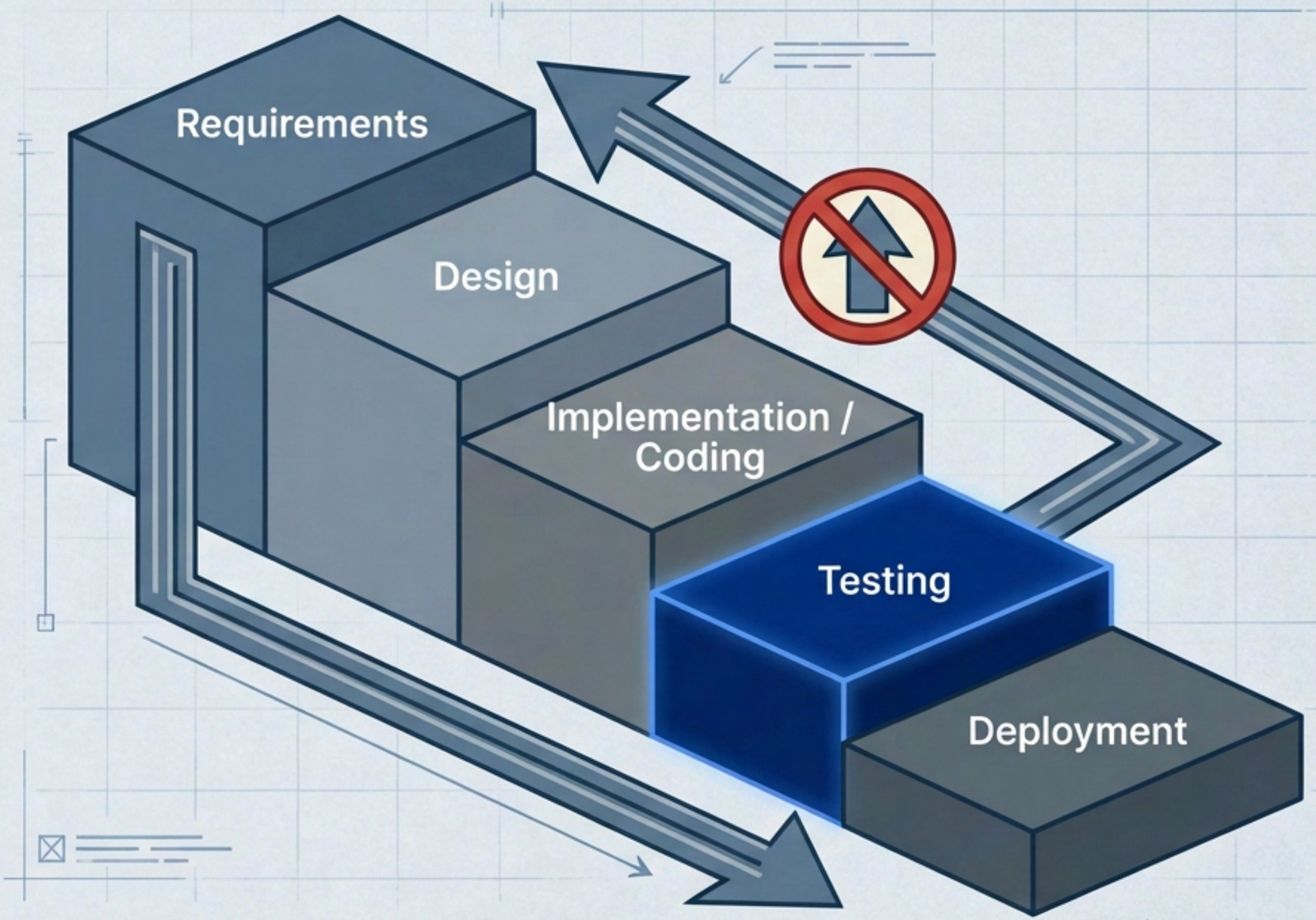
Waterfall: The Linear Cascade

Concept Definition

Waterfall is a sequential approach where progress flows downwards like gravity.

A phase must be 100% complete and signed off before the next begins. You cannot easily flow back up.

If a requirement was misunderstood in phase 1, catching it in phase 4 is costly.



Life in the Waterfall: The Guardian of the Specification

Source of Truth

The Requirements Document is law. If software behavior differs from the doc, it is a bug—even if the software functions "better".

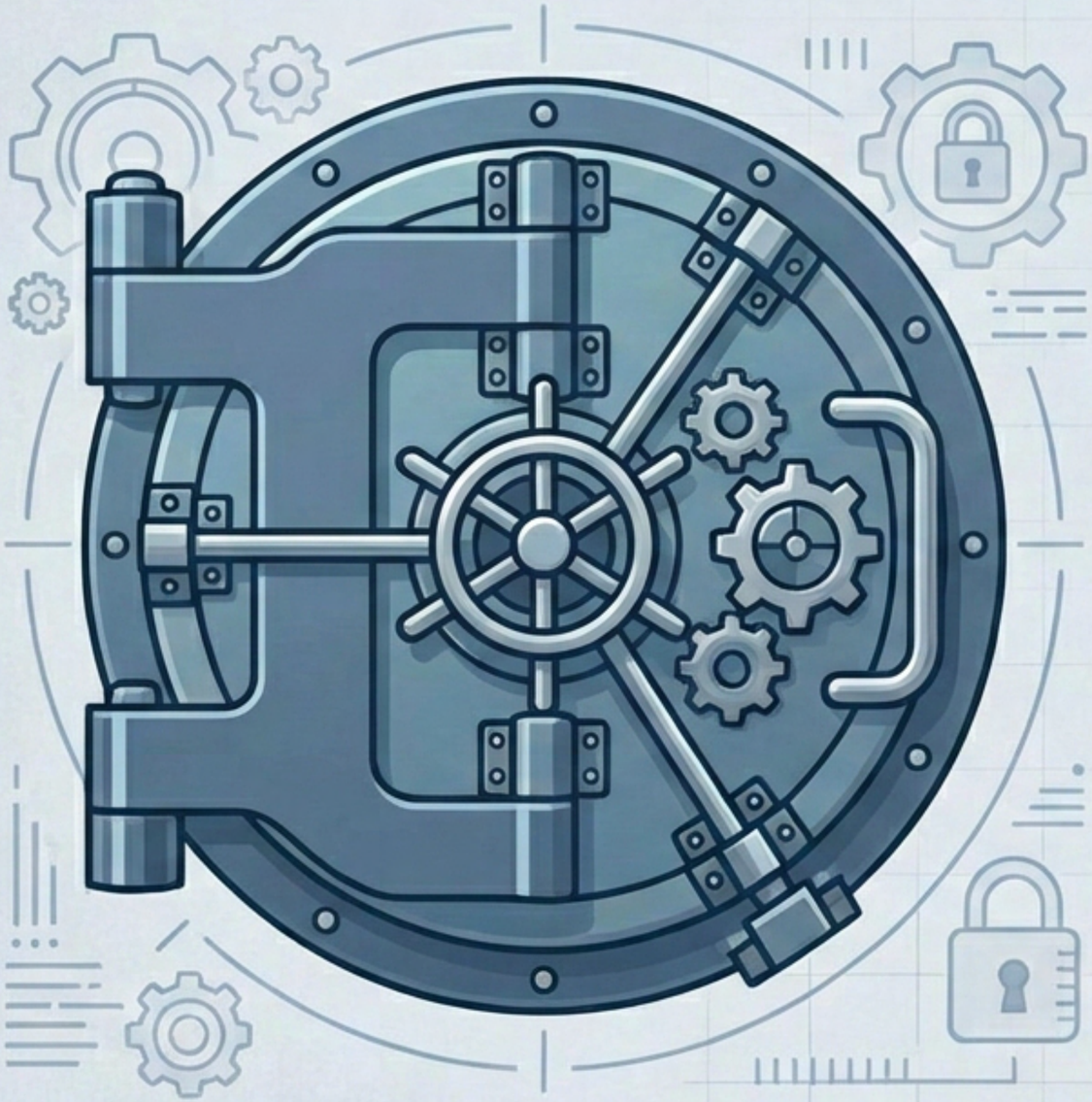
Key Artifacts

Formal Test Plans, Traceability Matrices (RTM), and detailed Test Cases with explicit pre-conditions and post-conditions.

The Reality

The Gatekeeper. You review requirements for testability and plan strategy long before seeing a single line of code.

Scenario: The Core Banking Update



The Mission

You are testing a mainframe update for a bank.

Why Waterfall Wins Here

Regulations require a strict paper trail for every change. Errors can cost millions or result in lawsuits. Stability is prioritized over speed.

The Tester's Experience

You spend 4 weeks writing test cases based on a 200-page specification document before you ever see the software. Precision is paramount.

The Trade-off: Predictability vs. Agility

The Pros

+ Clear Expectations: Expected results are documented; no guessing.

+ Tracking: Easy to measure progress via the Traceability Matrix.

+ Scope Control: No ‘scope creep’; requirements are stable.

The Cons

– The ‘Crunch Time’: Development delays often eat into the testing time, forcing rushed work at the end.

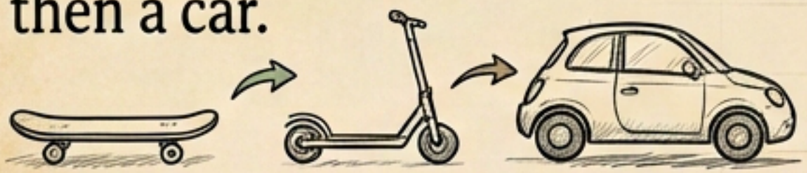
– The Wait: Boring periods during the coding phase.

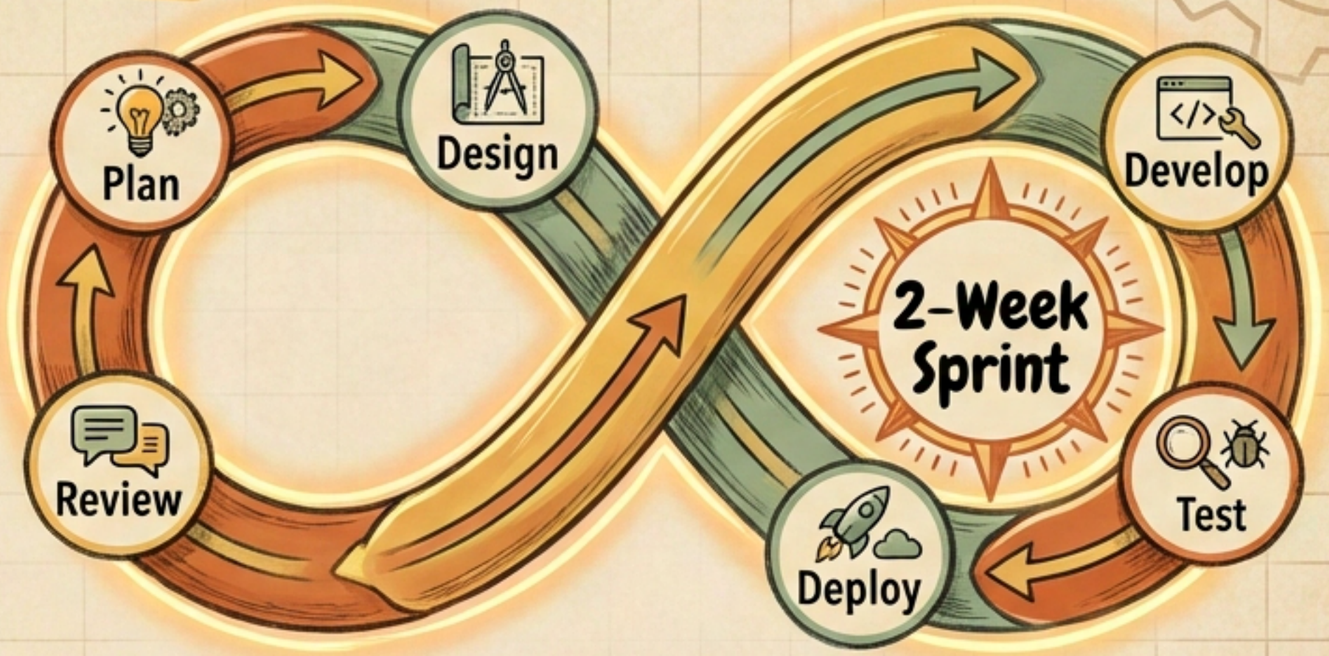
– Cost: Bugs found this late are expensive to fix because the code was written months ago.

Agile: The Iterative Loop

Concept Definition

Agile is iterative and incremental. Rather than building a whole car at once, you build a skateboard, then a scooter, then a car.





User Stories: Small, user-focused requirements.

Acceptance Criteria: Conditions to pass the story.

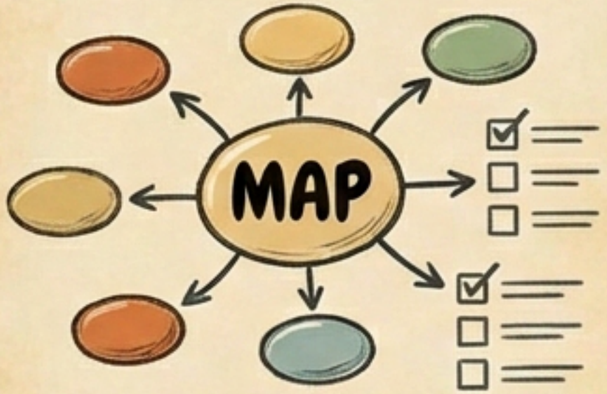
Definition of Done: Checklist for completion.

Life in Agile: The Quality Coach & Collaborator

Source of Truth

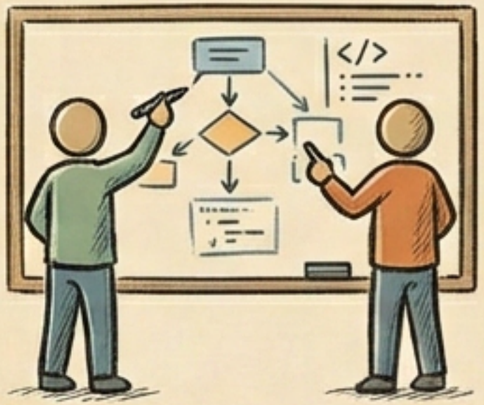
The User Story and ongoing conversations with the Product Owner. Documentation is fluid.

Key Artifacts



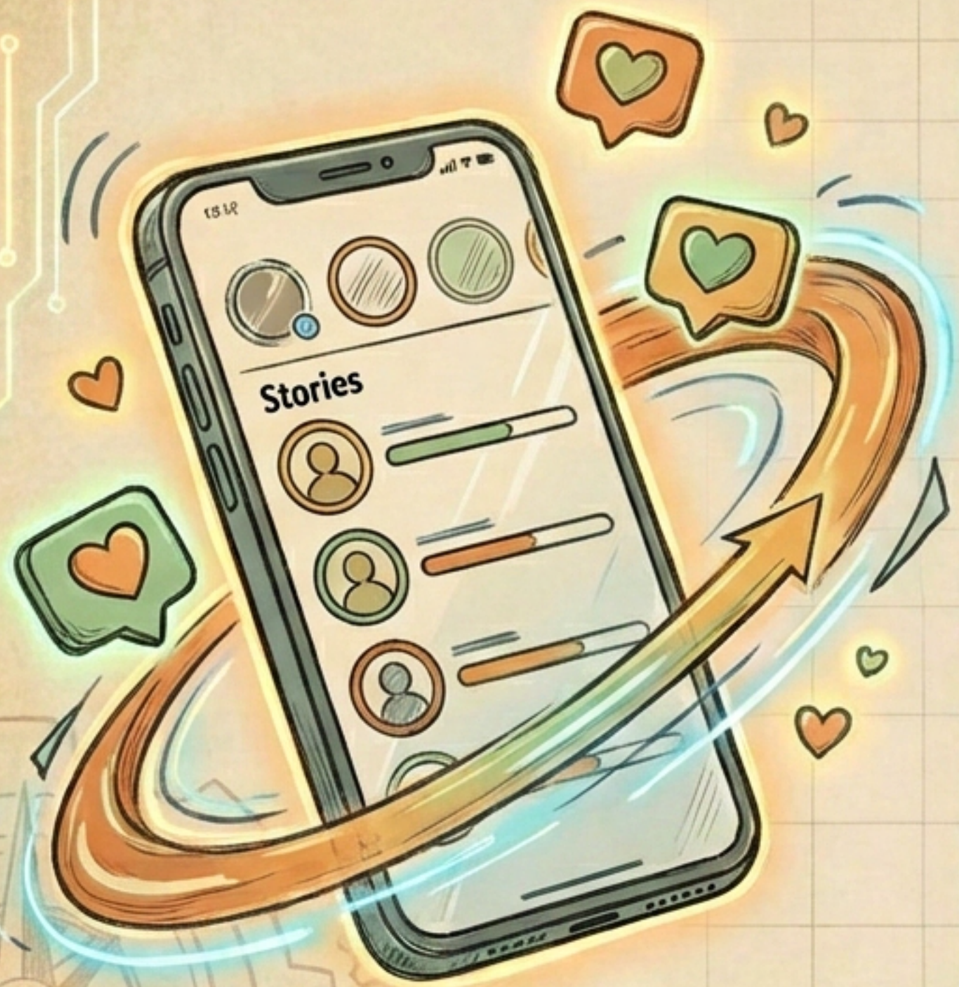
Lightweight tools like Mind maps, checklists, one-page strategies, and automated regression scripts.

The Reality



The Collaborator. Testing happens concurrently with coding. You might test a feature on Tuesday that was coded on Monday.

Scenario: The Social Media Feature



The Mission

Launching “Stories” to beat a competitor next week.

Why Agile Wins Here

- Market trends move fast. The company must release immediately.
- Requirements might change mid-stream based on user feedback.

The Tester’s Experience

- No heavy specs. You talk to the dev on Monday, agree on functionality, and test on Tuesday.
- You rely on exploratory testing to move fast.

The Trade-off: Speed vs. Documentation

The Pros

+ Immediate Feedback: Bugs are found instantly, making them cheap to fix.

+ Influence: Testers influence design early in the process.

+ Collaboration: Less loneliness; constant team interaction.

The Cons

- Regression Hell: You must re-test old features every 2 weeks (requires automation).

- Documentation: Often scarce, outdated, or non-existent.

- Instability: Requirements can change in the middle of a sprint.

At a Glance: Waterfall vs. Agile

Feature	Waterfall	Agile
Requirement Stability	**Fixed.** Change Requests needed.	**Flexible.** Changes welcomed.
Testing Timing	Distinct phase after coding.	**Continuous** during coding.
Tester's Mindset	**Verification** (Matches spec?)	**Validation** (Useful to user?)
Documentation	**Heavy, formal,** approved.	**Lightweight,** "Just enough".
Team Structure	**Silos** (Testers with Testers).	**Cross-functional** (Testers with Devs).
Feedback Loop	**Long** (Months).	**Short** (Days/Weekly).

Same Techniques, Different Application

Example: Boundary Value Analysis

Waterfall Application (The Formal)

Test ID	Description	Input Value	Expected Output	Traceability Link	Status
Test 01	Map every input combination from the Spec.	0	Expected Output	Traceability	✓
Test 05	Create 50-step test cases for min/max values.	500	Expected Output	Traceability	✓
Test 02	Given the user is on the login page	100	Expected Output	Traceability	✓
Test 003	Given they enter valid username and password	1100	Expected Output	Traceability	✓
Test 004	When they enter valid username and password	1300	Expected Output	Traceability	✓
Test 005	Given they enter valid username and password	1500	Expected Output	Traceability	✓
Test 006	When they enter valid username and password	200	Expected Output	Traceability	✓
Test 007	When they enter valid username and password	290	Expected Output	Traceability	✓
Test 008	When they enter valid username and password	367	Expected Output	Traceability	✓
Test 009	Given they enter valid username and password	309	Expected Output	Traceability	✓
Test 007	Then they enter valid username and password	780	Expected Output	Traceability	✓
Test 100	Then they os redirected to the dashboard	21	Expected Output	Traceability	✓

- **Map** every input combination from the Spec.
- Create **50-step test cases** for min/max values.
- **Link** every test ID via Traceability Matrix.

Agile Application (The Rapid)

```
Feature: User Login
  Scenario: Valid credentials
    Given the user is on the login page
    When they enter valid username and password
    Then they are redirected to the dashboard
```

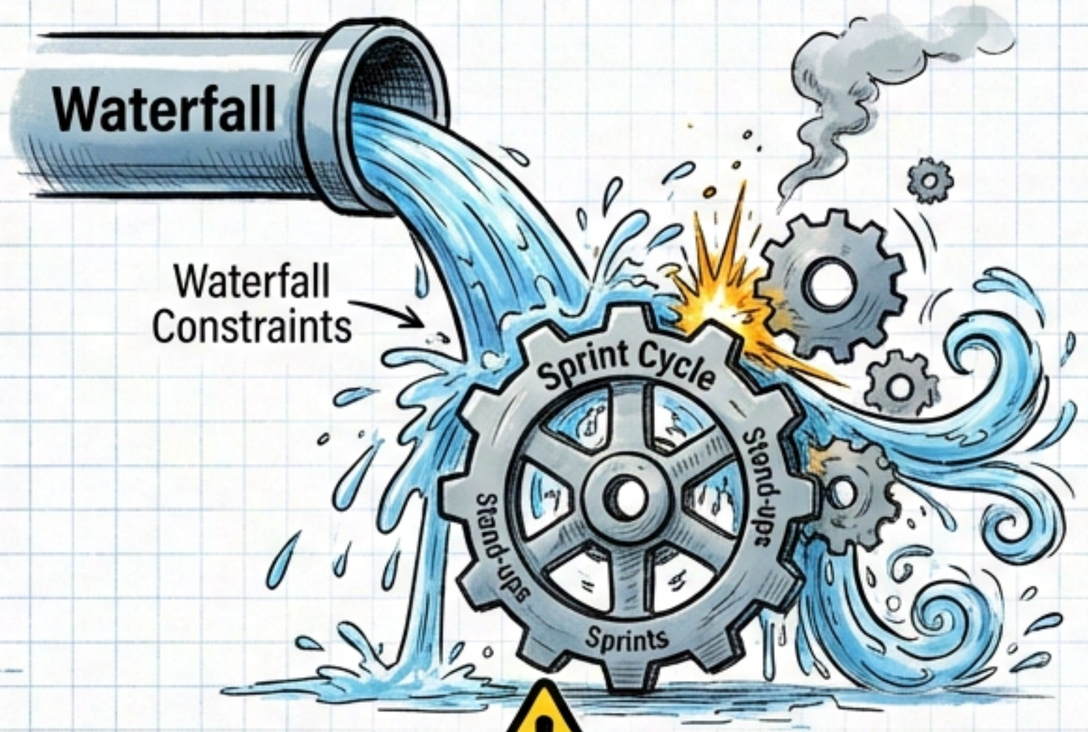
- **Exploratory Testing**: Use domain knowledge to ‘hunt’ bugs.
- **ATDD**: Write ‘Given/When/Then’ scenarios before code is written.
- **Documentation:** Quick checklists or Acceptance Criteria.

Critical Thinking: Beware the “Water-Scrum-Fall”

⚠ WARNING

The Trap

Many modern companies claim to be Agile but actually practice Water-Scrum-Fall.



The Symptoms

They hold Agile meetings (stand-ups, sprints) but maintain Waterfall constraints (fixed requirements, heavy deadlines, no flexibility).

The Advice

A **good tester adapts** to the team, not just the label. Focus on delivering value regardless of the methodology used.

Career Focus: Acing the Methodologies Interview

Which is better?

Depends.
Waterfall = Safety
Agile = Speed
(Consumer Apps).

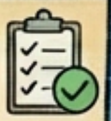
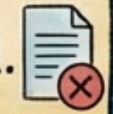


How do you test in Agile?

Shift left.
Less documentation,
more collaboration,
and automation.


Boundary Value in Agile?

Use the technique,
skip the 50-page doc.
Put it in the
Acceptance Criteria.

The Takeaway

Waterfall

- Control & Documents.
- You are the Gatekeeper.

Agile

- Speed & Conversation.
- You are the Quality Coach.

Master the technique (Black Box), then adapt the process (SDLC).