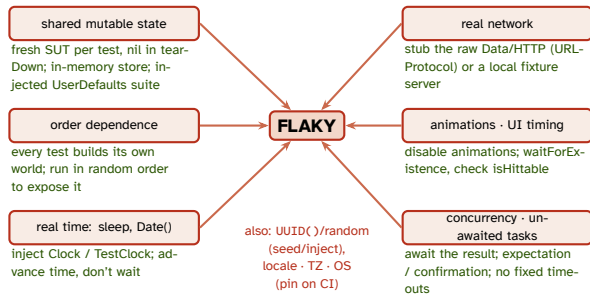


In one line: A flaky test has a **nondeterminism source** — shared state, order, real time, real network, animations, concurrency, environment — so **remove the source, never retry**. Coverage says what *ran*, not what was *verified*. TDD = **Red** → **Green** → **Refactor** in minute-long cycles; the failing test comes first.

Picture — flaky cause → fix



How it works

- **Random order + parallel** (test plan, or -parallel-testing-enabled YES): Xcode spreads test classes over simulator clones. A suite must pass in *any* order — if randomizing flips results you have hidden coupling. Last resort: @Suite(.serialized).
- **Retry is not a fix** — a flake is often a real race users also hit; quarantine with a ticket, then root-cause.
- **Coverage** = % of lines/branches *executed* (LLVM profdata). Enable: scheme → Test → Options → Gather coverage, or test plan; read with `xcrun xccov view --report R.xcresult`. **Branch** coverage beats line coverage (a guard's else path). Use it to *find untested risk*, watch the trend; don't gate on a % (teams aim ~70–85% on logic modules).
- **CI:** `xcodebuild test -scheme App -destination 'platform=iOS Simulator,name=iPhone 15,OS=17.5' -resultBundlePath R.xcresult -enableCodeCoverage YES` (or `fastlane scan`). **Pin** simulator *and* OS; keep the .xcresult, screenshots, snapshot diffs as artifacts; fast PR plan + slow nightly plan (locales × sanitizers).
- **TDD three laws** (Uncle Bob): (1) no production code without a failing test; (2) no more test than enough to fail — not compiling counts; (3) no more production code than enough to pass. **Fake it** → **triangulate** (a 2nd example forces the general rule). **Refactor only on green**; refactor = behaviour-preserving.
- **BDD:** same loop, phrased as behaviour — **Given / When / Then** (= Arrange/Act/Assert); Quick/Nimble, Cucumber/Gherkin. **London** (outside-in, mockist) vs **Detroit/Chicago** (inside-out, classicist).
- **Test-induced design damage** (DHH vs Beck/Fowler, “Is TDD dead?” 2014): bending code *only* for tests — one-impl protocols, exposed internals. Counter: damage comes from over-mocking; testability usually = good decoupling.

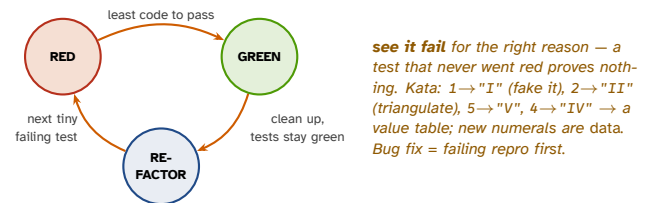
Example — kill a time flake

```

// FLAKY: real clock -- 2 s slow, fails on a loaded CI box
func test_expires_bad() {
    let t = Token(ttl: 1); sleep(2); XCTAssertTrue(t.isExpired) }

// DETERMINISTIC: inject the clock, move time yourself
protocol Clock { var now: Date { get } }
final class TestClock: Clock {
    var now = Date(timeIntervalSince1970: 0) }
func test_expires_afterTTL() {
    let clock = TestClock()
    let t = Token(ttl: 60, clock: clock)
    clock.now += 59; XCTAssertFalse(t.isExpired) // boundary
    clock.now += 2;  XCTAssertTrue(t.isExpired)  // instant
}
  
```

Red → Green → Refactor



Interview traps (the catalogue)

- ! **Tautology:** stub returns `User(id:1)`, assert `User(id:1)` — stub the *raw Data* so your parsing runs.
- ! Testing the framework (JSONDecoder works) instead of your mapping.
- ! Asserting private state / partial-mocking the SUT → refactor-fragile.
- ! Over-mocking: all green, prod broken — fakes lie. Add contract + integration tests.
- ! Ordered == on Set/Dictionary output → sort or compare sets.
- ! No assertion = inflates coverage, tests nothing. 12 asserts in one test = poor localization.
- ! Test written after the fix: revert the fix — if it doesn't go red, it proves nothing.
- ! “100% coverage” mandate → assertion-free tests, punished deletions.

Remember

Control time, don't wait on it. Isolate state · stub I/O · pin env · random order. Coverage finds gaps, not quality. **Fail first, then pass, then clean.**

Likely questions

1. Top flaky causes? — shared state, order, sleep/time, network, animation, concurrency.
2. Passes locally, fails on CI? — locale/TZ, simulator/OS, parallel coupling, speed.
3. Is 100% coverage good? — executed ≠ verified; use branch coverage on risk.
4. Three laws of TDD? — failing test first; just enough test; just enough code.
5. When does TDD fit badly? — exploratory UI/animation, spikes, thin glue.