

Flaky tests: from 0.5% to 98%

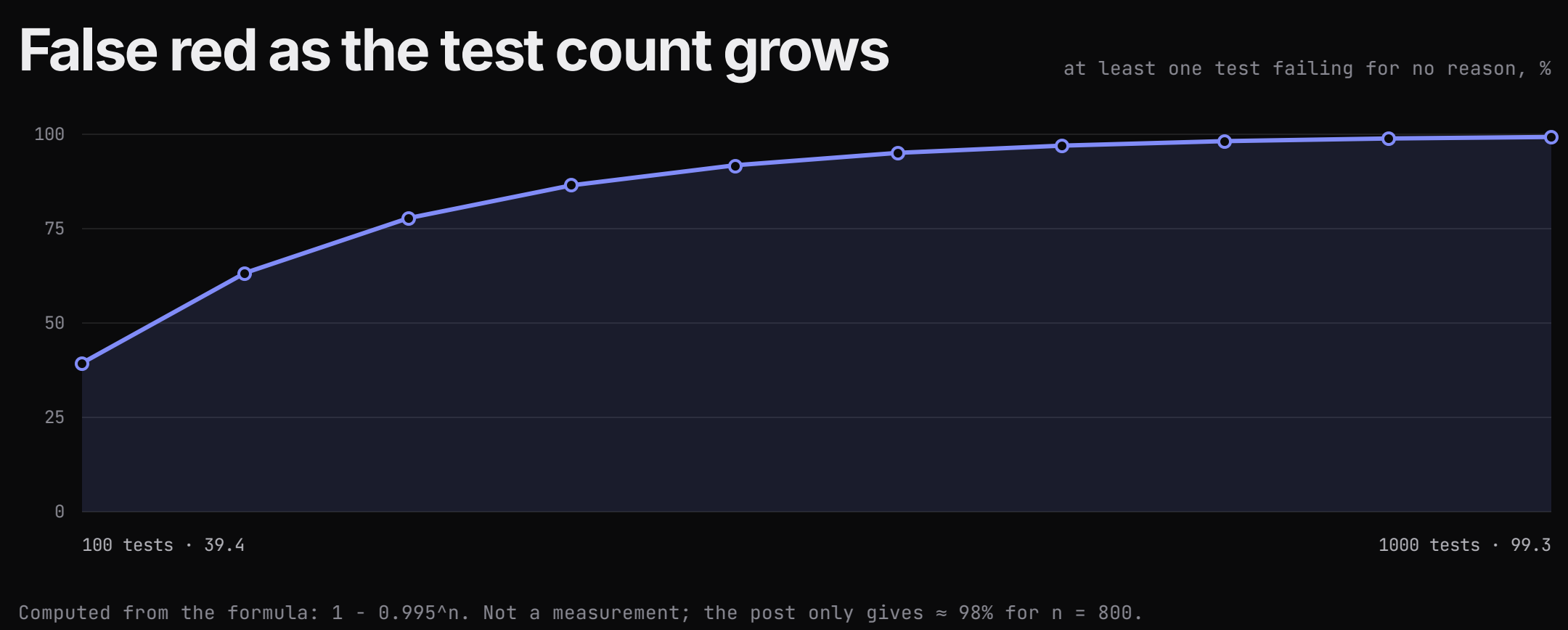
If each of 800 tests fails for no reason 0.5% of the time, the suite is red on almost every run. Why retry isn't a cure, and what catches flakiness?

01 THE BIGGER THE SUITE, THE LESS RELIABLE

98%

Chance that any test fails on a clean run

A single test that fails for no reason 0.5% of the time is invisible. Put 800 of them together: $1 - 0.995^{800} \approx 98\%$.



03 FAILURES AREN'T INDEPENDENT EITHER

50 ms locally, a timeout under load.

CI runs on shared, throttled hardware; an async test that finishes in 50 ms on your laptop falls into a timeout under load.

04 AI DIDN'T INVENT FLAKINESS, IT MASS-PRODUCED IT

Four patterns that look right

⚠

A fixed sleep

To "wait" for async work:
time.Sleep(2 * time.Second),
await
page.WaitForTimeout(2000),
cy.wait(2000).

⚠

Unseeded randomness

A different value on every run:
Math.random(), rand.Intn(100), a
faker with no fixed seed.

⚠

A brittle selector

Nailed to the DOM structure:
nth-child, an auto-generated
class name. Breaks when the
markup shifts a little.

⚠

A mock for everything

The test verifies an interaction,
not an outcome.

05 RETRY IS A TREADMILL, NOT A CURE

All of it is reactive

✖

Retry

Burns real CI minutes and
stretches the feedback loop;
you pay twice to learn nothing
new.

✖

Quarantine

A silently growing pile of
disabled tests. That pile is debt,
and it compounds.

✖

Ticket

Ages in backlog noise and
quietly stops mattering.

06 CATCH THE ANTI-PATTERN, NOT THE FAILURE

When does it step in?

	Retry, quarantine, ticket	Static scan at the commit boundary
TIMING	After the test has already broken CI	✓ Before the test even runs, in the diff
EFFECT	Hides the symptom, sends you the bill	✓ Flags the known anti-patterns
REPEATABILITY	Passes on rerun, then fails again	✓ Same diff, same result; no model, no network
SCOPE	Doesn't make the test deterministic	✓ Stops obvious flakiness before it ever reaches CI

07 DETECTION IS THE SAFETY NET, DESIGN IS THE REAL FIX

Write the test deterministically

☐

Replace fixed waits with condition-based waits. Wait for a fact, not for the clock.

☐

Seed every source of randomness; inject a deterministic clock and ID.

☐

Select by role or test id, not DOM position.

☐

Mock the boundary, not the logic; keep at least one real integration path.

08 WHEN DOES THIS PATTERN STOP BEING THE RIGHT ANSWER?

Limits of the static gate

ⓘ

Won't catch a clever race

It catches the obvious anti-patterns, not a race condition buried inside your own code.

⚠

Keep the rules conservative

A noisy gate gets ignored; an ignored gate is worse than none.

ⓘ

If the system is flaky

A test that is flaky because the system under test isn't deterministic is telling you something about the system.

A green CI you have to rerun isn't green.

The cheapest place to kill a flaky test is the line it's written on: before it costs you a build.

On the line it's written

sade.dev

Source: sade.dev, Flaky Tests: Retrying Isn't a Fix

Muhammet Şafak · muhammetsafak.com/visuals//flaky-tests-98-percent-red/