TQS THE QUALITY SIGNAL
BY PANKAJ NAKHAT

Edition 01
MAY 2026
Monthly

LEAD STORY

AI is not replacing QA. It is replacing bad QA.

The teams panicking about AI-driven testing are the same teams whose regression suites take 4 hours to run and catch nothing. The teams thriving? They built signal-first quality from the ground up.

PN **Pankaj Nakhat**
Head of QA, Release & Reliability · Real Estate Tech

// FEATURE

Why your CI pipeline is lying to you every single day

Here is a scenario I see repeated across fintech and real estate tech, inside every organisation that scaled fast. The pipeline goes green. The release ships. The incident lands in production three hours later. The post-mortem finds the defect was there the whole time, hiding in a test suite that never ran the right scenario against the right data under the right conditions.

Your pipeline is not a quality gate. It is a confidence theatre. Passing tests do not mean working software. They mean the scenarios your team thought to write six months ago still behave the way they did six months ago. That is a different thing entirely.

> *"A green build is evidence that your known risks are managed. It tells you nothing about your unknown ones."*

The shift I am seeing across high-performing engineering organisations is a move from coverage metrics to signal metrics. Not "how many tests do we have" but "how quickly does a real defect surface in our pipeline." Not "what is our pass rate" but "what is our mean time to detection."

4x

faster MTTR in teams using signal-first quality practices

68%

of production incidents were detectable in pre-prod with the right observability

23min

average CI runtime in elite DORA performers vs 4h+ in low performers

Three things separate organisations that catch defects early from those that find them in production. First, they instrument for signal before they write tests. Second, they treat flaky tests as P1 incidents, not background noise. Third, they close the feedback loop between production telemetry and their test strategy within a sprint cycle.

In one of the largest real estate tech estates I have worked in, running 25+ product teams, the change that moved the needle most was not adding more tests. It was killing the ones that lied. We cut 30% of our regression suite in one quarter and our defect leakage rate dropped. Because we stopped measuring test quantity and started measuring detection velocity.

```
// signal-first quality: the metrics that actually matter
detectionVelocity: "time from defect introduction to pipeline alert"
signalRatio: "failing tests that led to real fixes / total failures"
flakyTaxRate: "engineer hours lost to noise per sprint"
leakageRate: "defects found in prod / defects found pre-prod"
```

If you are not tracking these four numbers, you do not know the quality of your quality process. You just know how many tests you have.

// QUICK SIGNALS

Four things worth your attention this month

AI TESTING

Playwright's native healing layer changes everything

v1.56 ships with agentic Planner and Healer agents. The era of brittle locators is ending. The era of explainable test failures is beginning.

DORA 2025

Elite performers deploy 973x more frequently

The gap between elite and low DORA performers widened again. The differentiator is not tooling. It is psychological safety in the release process.

FINTECH QA

OAuth2 and API testing is the new regression suite

ForgeRock, Okta, Auth0 integrations are where 40% of production incidents originate in financial services. Your UI tests are testing the wrong layer.

LEADERSHIP

Your QA team should own the release decision

If QA is a gate that PMs push through, you have already lost. Quality is a shared outcome, but someone needs to hold the signal. That is QA leadership's job.

// WHAT I AM WATCHING

The five-agent QA architecture is becoming production reality

Multi-agent test orchestration is moving from research paper to production workflow in 2026. Single-agent LLM interactions are limited by context window size, inability to parallelise, and lack of specialisation. The architecture that solves this decomposes QA into five specialised, collaborating agents.

AGENT ARCHITECTURE	
Analyst	Reads requirements, produces structured test plan JSON. Uses chain-of-thought to reason through coverage gaps before outputting. Nothing moves to Writer without this output.
Engineer	Consumes the approved test plan, generates executable Playwright/Supertest code. Few-shot examples of your team coding style are the entire difference between useful and generic output.
Sentinel	Runs tests, captures results, classifies failures. Its single job: distinguish a product bug from a test bug from a flaky environment. It does this before a human reads the output.
Healer	Detects broken locators, proposes minimal-diff fixes. Strict scope: it touches only locators, never assertions or test logic. Every proposed fix outputs as a unified diff. Human reviews first. Always.
Oracle	Synthesises results into executive quality reports. Structured JSON output only, conforming to a defined schema. The Oracle writes the go/no-go recommendation. The QA lead owns the decision.

The four gates that keep humans in the loop:

- **Spec Review Gate** - Analyst output reviewed by QA Lead before Engineer runs. No automated pass-through.
- **Diff Review Gate** - Every Healer fix reviewed before merge. Diffs are never auto-applied.
- **Data Gate** - Any test that creates or modifies production-adjacent data requires explicit human approval.
- **Commit Gate** - Generated tests go to a feature branch, not main. PR review is mandatory, not optional.

The architecture works because it is not trying to remove humans from QA. It is removing humans from the parts of QA that are mechanical, repetitive, and low-judgment. The high-judgment work stays exactly where it belongs.

// PROMPT OF THE MONTH

The failure triage prompt that saves 25 minutes per CI failure

Manual CI failure triage is one of the highest-friction, lowest-value activities in modern QE. The same reasoning loop runs every time: read the error, check the stack trace, look at recent commits, form a hypothesis. This is exactly what a well-structured prompt handles reliably.

```
// A.3 — Failure Triage (from the QE Prompt Library)
ROLE: QA engineer triaging a CI test failure.

Test: [TEST NAME]
Error: [ERROR MESSAGE]
Stack: [STACK TRACE]
History: [LAST 5 RESULTS — PASS/FAIL pattern]
Commits: [RECENT COMMIT MESSAGES]

// Classify as exactly one of:
PRODUCT_BUG | TEST_BUG | ENVIRONMENT | FLAKY

// Output JSON only:
{ classification, confidence, rationale, nextAction }
```

The classification matters more than people realise. A PRODUCT_BUG triggers a Jira ticket automatically. A TEST_BUG goes to the Healer. An ENVIRONMENT failure pages the platform team. A FLAKY classification opens a stability ticket against the owning team. The same 30-line prompt replaces four different manual workflows, runs in under 10 seconds, and produces a structured output that feeds directly into your incident management system.

> *"The model settings matter as much as the prompt. Set temperature to zero for triage. Classification must be deterministic. There is no room for creative interpretation when you are deciding whether to page someone at 2 AM."*

// ANTI-PATTERN WATCH

Three prompting mistakes that cost teams the most this quarter

ANTI-PATTERN 01

Hallucination acceptance

Teams commit AI-generated tests without running them. LLMs confidently generate calls to methods, selectors, and imports that do not exist. The fix is non-negotiable: every generated test must execute before it is committed. Unexecuted AI code is untested code. Full stop.

ANTI-PATTERN 02

Context overload

Injecting entire codebases, full OpenAPI specs, and all existing tests into a single prompt. Output quality degrades sharply past a certain token volume, and cost spikes with no quality return. Extract only the minimum necessary context. Testing one endpoint? Inject only that endpoint's definition, not the whole spec file.

ANTI-PATTERN 03

Self-healing as a communication substitute

Using the Healer agent instead of fixing unstable selectors or talking to the engineers who own the component. Every self-heal event must trigger a notification to the owning team and a ticket to stabilise the selector at source. Self-healing buys time. It does not fix the problem. If it starts buying time repeatedly on the same locator, you have a different problem entirely.

// TOOL SPOTLIGHT

Open source worth trying

PS

playwright-spec-doc-reporter

Generates living documentation from Playwright runs, with traceability links, manual test support, and healing layer integration. Referenced in the enterprise `playwright.config.ts` pattern as a first-class reporter alongside HTML and JUnit. Built for teams that need audit-ready test evidence without manual overhead. Battle-tested across a 25-POD product estate, now open for any team that needs test output to tell a real story.

`npm install playwright-spec-doc-reporter ↗`

THIS MONTH: READING LIST

- **Prompt Engineering for Quality Engineering** — A practitioner guide covering 16 chapters, 50+ production-tested prompts, multi-agent architecture, Claude Code integration, and a full prompt governance framework. Free at pankajnakhat.com.
- **DORA 2025 State of DevOps Report** — The data behind the 973x deployment frequency gap. Read the primary source, not the summaries.
- **Playwright v1.56 Release Notes** — The Planner, Generator, and Healer agent integrations are not marketing. Read the actual API surface before forming an opinion.

"Quality is not a department. It is not a phase. It is the continuous, measurable reduction of surprise in production. Everything else is process theatre."

PANKAJ NAKHAT / THE QUALITY SIGNAL

The Quality Signal by Pankaj Nakhat
Head of QA, Release & Reliability · Real Estate Tech, Abu Dhabi
pankajnakhat.com

EDITION 01
MAY 2026