
Five Structural Problems of Large-Scale Software in Industry

Emmanuel Zarpas, SAP & LIP6

Séminaire MoVe — LIP6

3 avril 2026

Context

- SAP · 100K+ employees — building systems for core business processes (ERP, analytics, data)
- R&D Analytics, Levallois — financial planning software
- Current work: prediction intervals for financial time series (bootstrap methods)
- CIFRE PhD in the team, co-supervised by Pierre-Henri Wuillemin (LIP6)
- Previous work: Continuous Integration trace analysis for developer experience → this talk

The Observation Ground

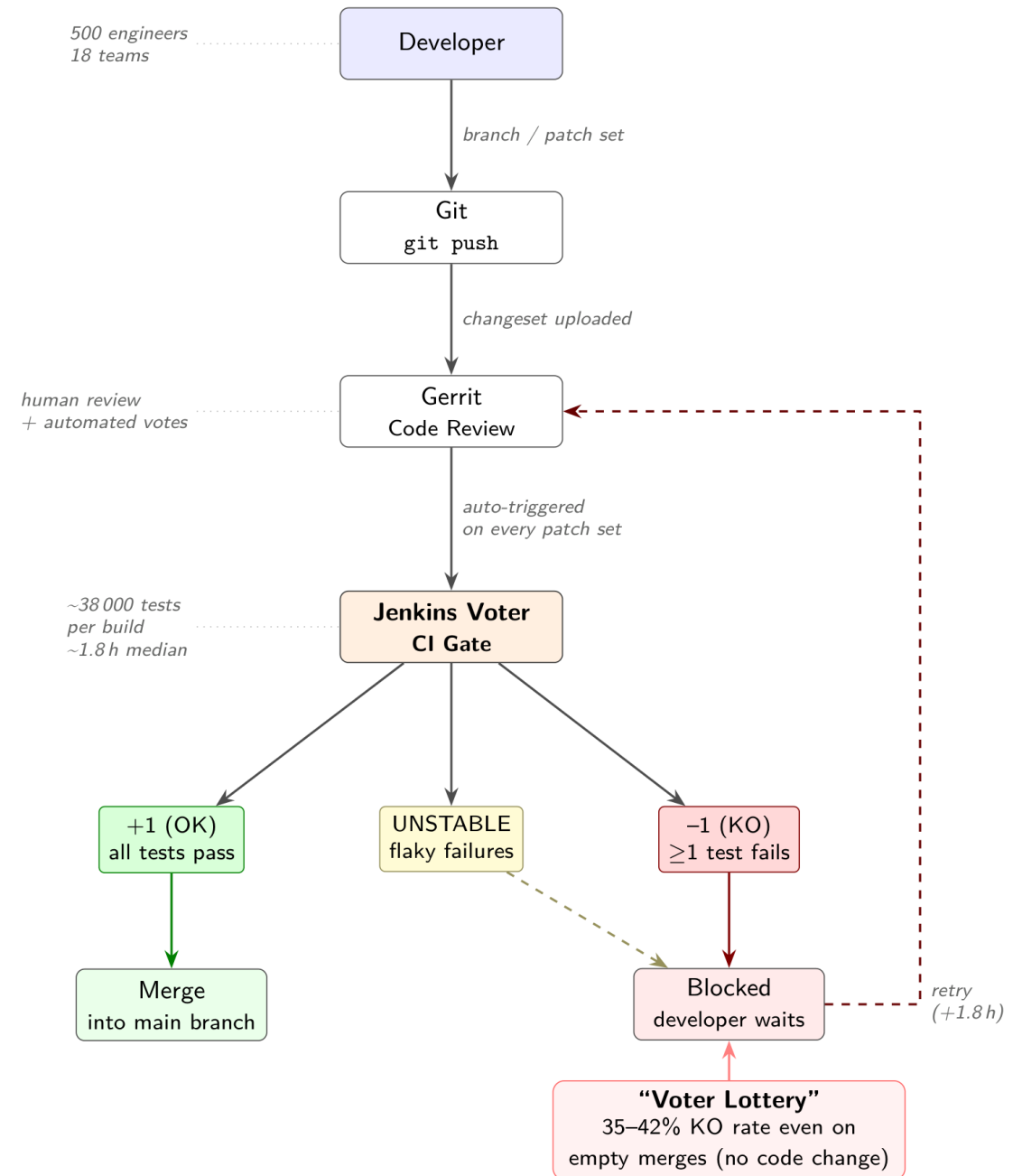
- One codebase as illustration
- Enterprise cloud product — cloud-native, organic growth
- ~500 engineer developers, 3 continents (North America, EU, India, China)
- Central main repository, 18 teams / branches
- Year 7: codebase and team grew — debt compounded
- CI (Continuous Integration) perceived as hindrance, not safety net

Scope of the Analysis

- 8-month embedded observation (Jun 2022 – Feb 2023)
- Continuous integration server (Jenkins) logs: outcomes, durations, test-level XML
- Code-review gateway (Gerrit): authors, branches, file lists
- Derived: builds $\leftrightarrow$ changesets cross-reference
- 72,000 builds · 530,000 distinct test IDs
- 2.5 billion individual test results stored

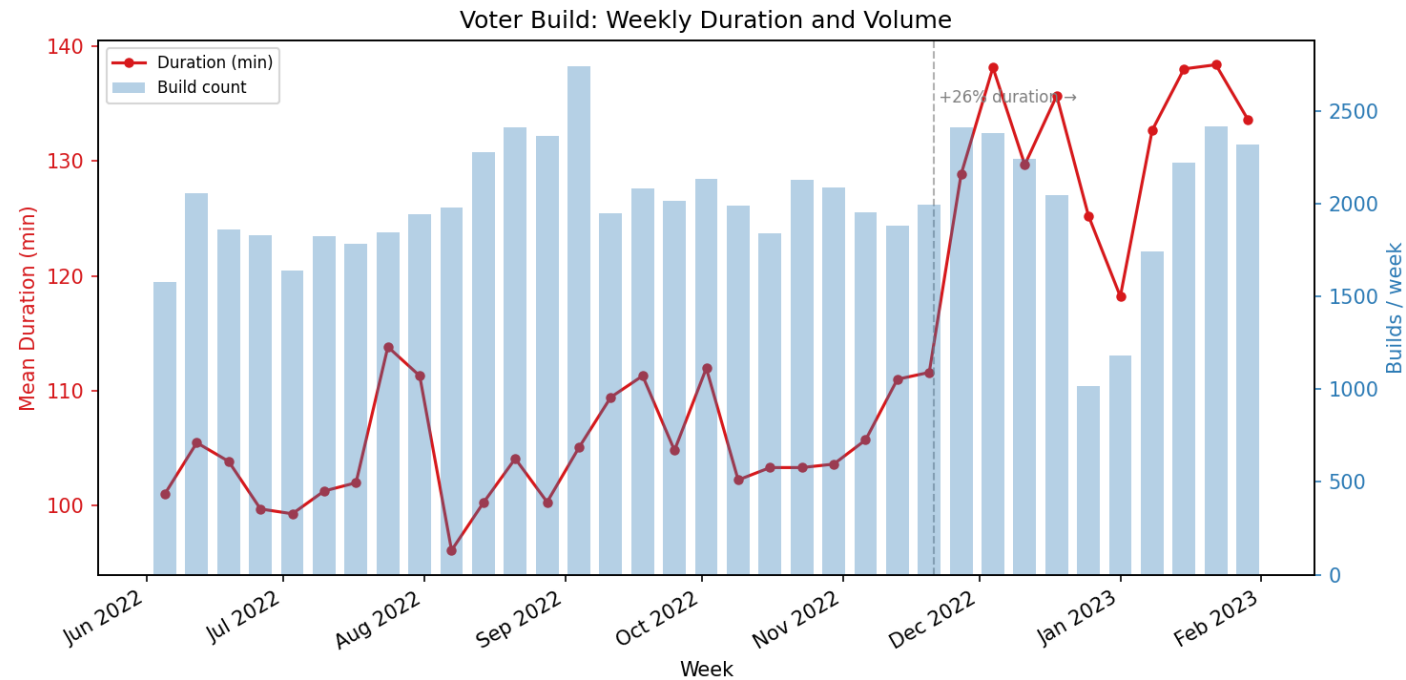
The Voter Lottery

- CI gate ("voter"): every commit must pass before merge
- Nearly 2 hours per build; ~38K tests (branch) or 100K+ (full)
- 88% of full runs have at least one failing test
- 35% fail (KO = build rejected) even on empty merges (zero code change) → a lottery



76 Million Tests a Week

- 2K builds/week × 38K tests ≈ 76M /week
- 3,800 CI-hours/week · 2.5B rows in 8 months
- Duration: ~1.8 h median → 2.2 h late in period

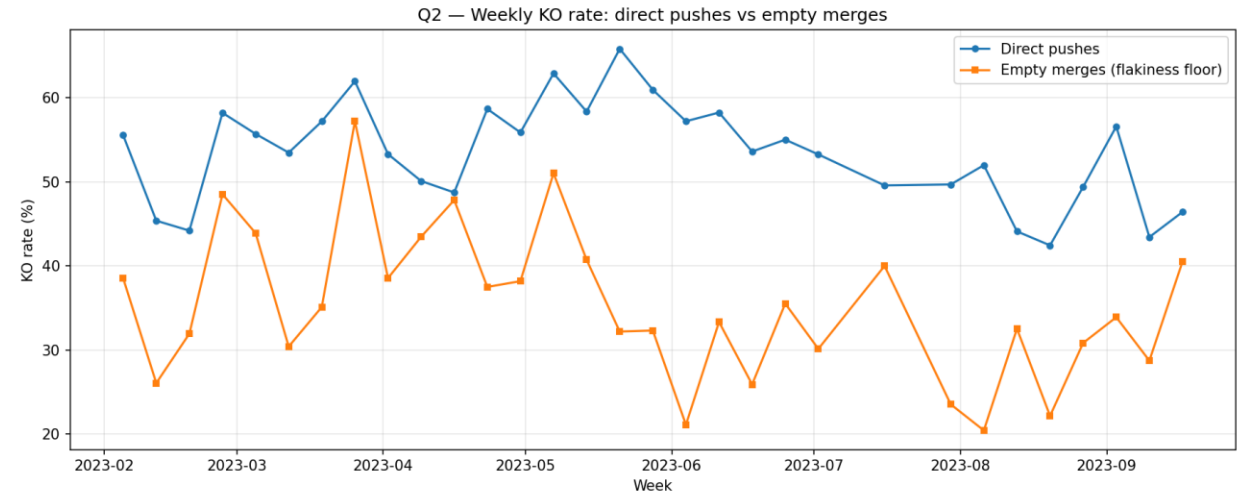


1. The Limits of Testing as a Proxy for Quality



The Noise Floor

- 35% KO on empty merges (zero code change)
- 0.08% test failures → yet 55% of builds fail
- KO verdict has low diagnostic value



The Pareto of Test Failure

- 1.4% of tests (7,600/526K) → 50% of all failures
- Integration tests with remote datawarehouse: 3% failure rate (37× average)
- Query-model tests (analytical queries): 0.8% (10× average)
- All tests: 0.08% failure rate

Non-Discriminating Tests

- 526K distinct test IDs over 8 months
- 74% never fail once (389K tests — zero diagnostic signal)
- 192K ephemeral IDs (seen once, then gone)
- 4,300 always-skipped (disabled when present)
- Effective test suite $\ll$ nominal test suite

Failure Hotspots

- 21 source files above 60% KO
- 604 files at 100% KO (≥ 5 builds)
- Known problems — "refactoring is on the roadmap"

File	KO rate	Avg commit size
UI5VersionSpec.js	83%	18 files
jest.config.js	70%	52 files
VizWidgetSpec.test.tsx	67%	19 files
VisualizationControl.js	62%	110 files
FireflyManager.js	63%	38 files

Noise Dominates Signal

- Changeset features cannot predict outcome
- No predictive size effect: 1-file commits 50% KO, 2-files 57%, 3-files 49%, 100 files 62% KO
- Weekly KO swing: 20% → 57% on empty merges
- Volume-independent: constant failure proportion

Synthesis

No feature predicts build outcome — the voter is a lottery

- $0.08\% \times 38\text{K tests} \approx 30 \text{ failures/build average}$
- Tests KO for KO build KO: median = 4, mean = 68
- Signal-to-noise ratio too low

2. Coordination Across Large Teams



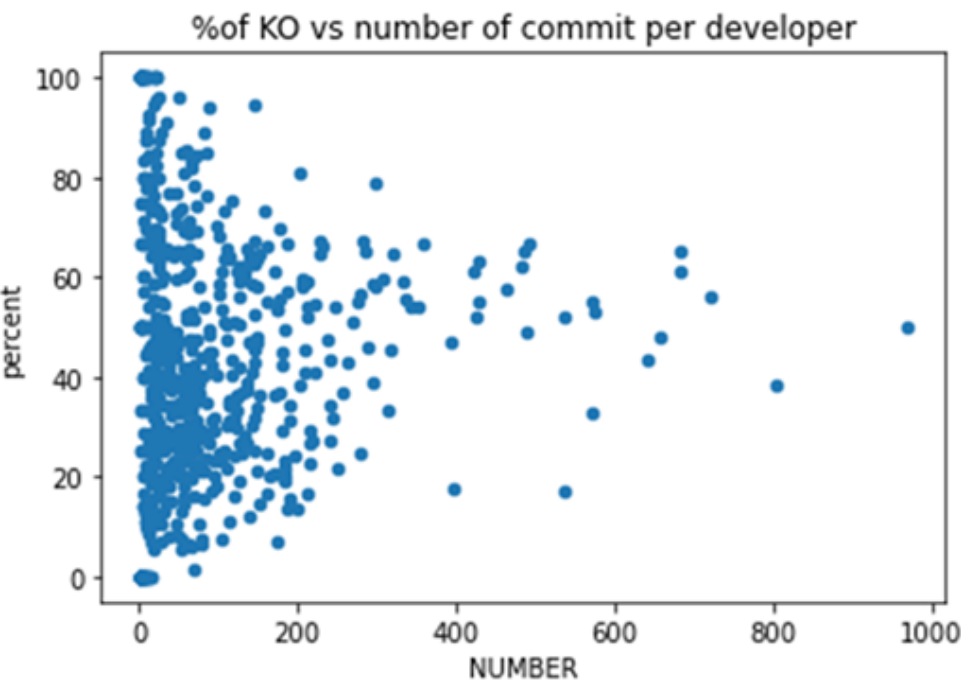
The Team Determines Your Fate

- 8% → 92% KO rate across 18 teams
- Same CI, same codebase — 11× spread in failure regime
- Activity does not explain outcome

	GERRIT_BRANCH	NUMBER	ko	percent
0	workstream_app_building	107	98	91.59
1	workstream_app_deploy	249	214	85.94
2	workstream_app_building	786	630	80.15
3	workstream_app_deploy	473	368	77.80
4	workstream_app_building	22636	14325	63.28
5	workstream_app_deploy	358	223	62.29
6	workstream_app_building	449	267	59.47
7	workstream_app_deploy	5651	3234	57.23
8	workstream_app_building	6604	3319	50.26
9	workstream_app_deploy	1576	723	45.88
10	workstream_app_building	3952	1205	30.49
11	workstream_app_deploy	3145	934	29.70
12	workstream_app_building	13793	4033	29.24
13	workstream_app_deploy	2971	650	21.88
14	workstream_app_building	3762	745	19.80
15	workstream_app_deploy	2492	436	17.50
16	workstream_app_building	1458	235	16.12
17	workstream_app_deploy	1255	99	7.89

Four Developers, One System

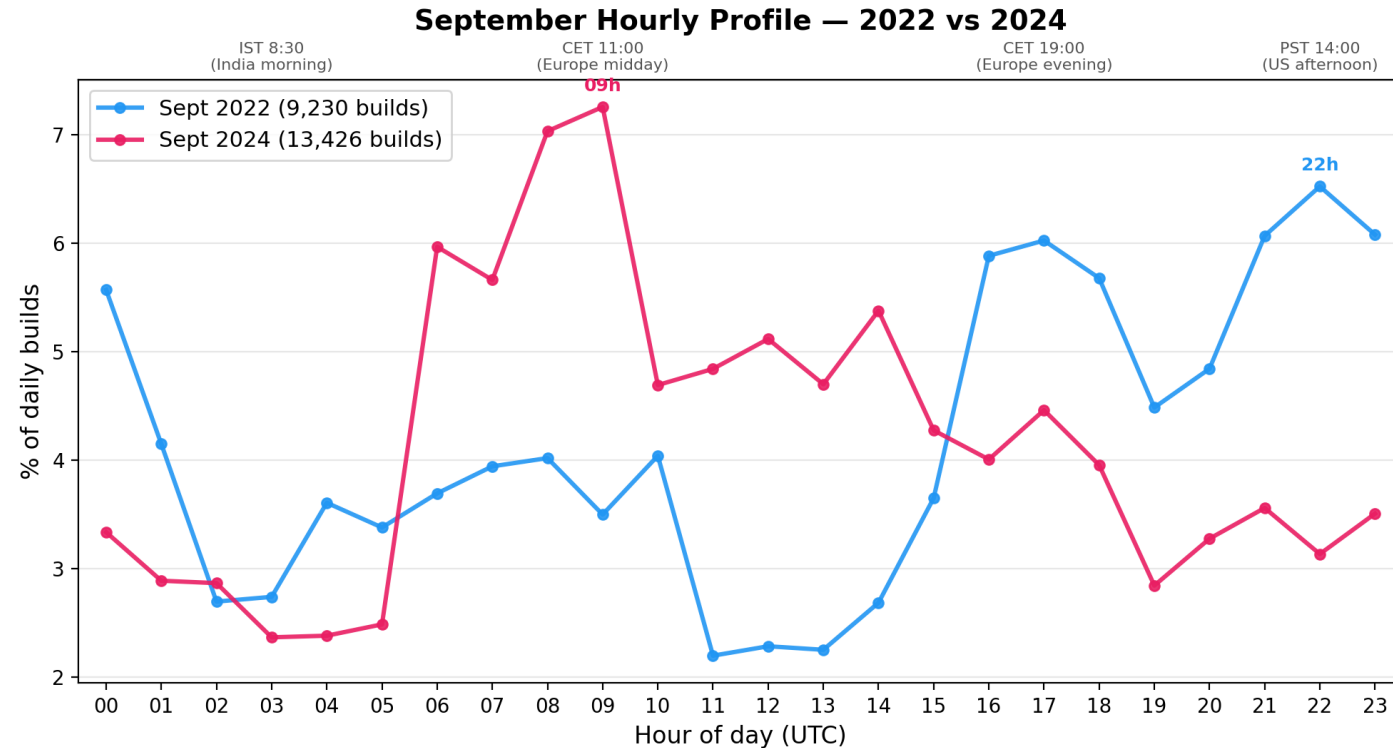
- 18 teams explain 14% of outcome variance; 682 individuals: also 14%
- Local testability > individual skill



Dev	Role	Commits	KO%	Archetype
Thomas	DevOps	20	100%	Phantom committer
Seb	Architect	26	11.5%	Isolated specialist
Fabrice	Senior UI	55	60.0%	Structural victim
Benoit	Developer	147	62.6%	Firefighter

The CI Never Sleeps

- Sept 2022: peak at 22h UTC (15h Vancouver time, 2h Paris time)
- Sept 2024: peak shifts to 09h UTC (14h30 Bangalore time); volume doubles
- Org restructuring becomes a CI time signature



Missing Observability

- From time zones to files: drift is visible, not monitored
- Some source files committed 32× during experiment — 0 OK, no alert
- 47% of source files safe (for 8 months) hit 80%+ KO later on
- No automated detector, no circuit-breaker, no formal escalation

The One-Way Trap

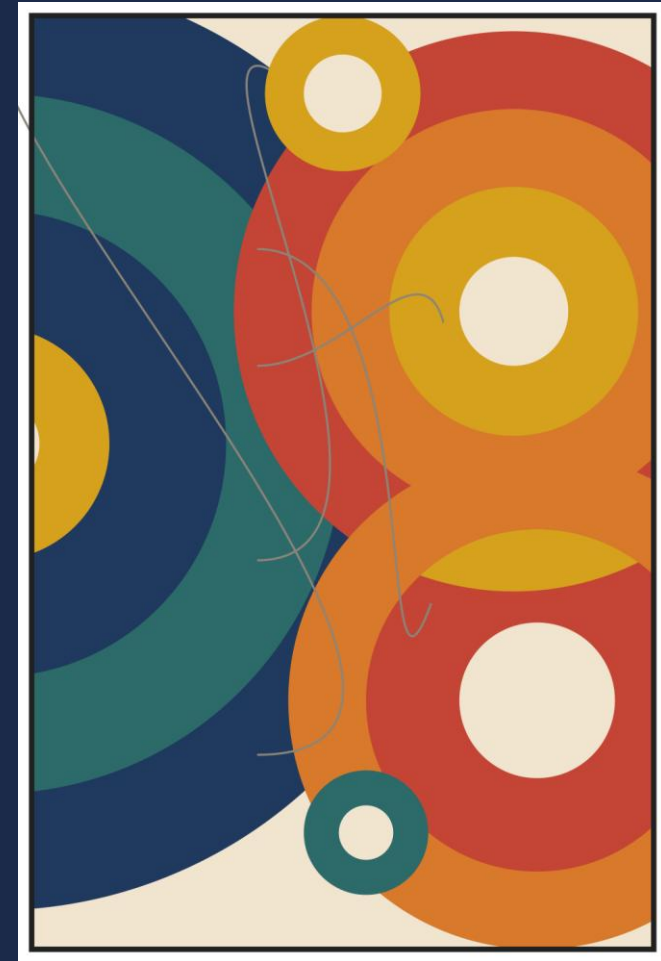
- Safe → KO: of 248 continuously active safe source files, 47% reached $\geq 80\%$ KO later on — none stayed fully safe
- Exit from KO is rare: 66% stay $\geq 80\%$ KO, only 5% recover below 50%
- Degradation behaves like an absorbing state

Synthesis

Conway's Law governs CI: team structure explains failure and degradation goes undetected

- 18 team labels predict failure as well as 682 individual identities
- Organization restructuring shifts CI patterns
- Cross-team, cross-continent coordination becomes delayed and fragile
- Files that start failing rarely recover — and no detector flags it
- drift is visible but unmonitored

3. Certification Requirements as Process Overhead



Why Certification Creates Overhead

- ISO 9001, ISO 27001, GDPR: recurring proof burden and constraints
- They require traceable evidence, not only test execution
- CI produces 76M executions/week — seldom linked to a requirement
- Adding a test: 1 commit. Removing one: who proves it's safe?

The Test Suite Can Only Grow

- 526K test IDs in 8 months — 192K seen once, 4,300 never executed
- Coverage metrics requires volume not usefulness
- Entry gate. No exit gate. No ownership. No retirement policy.
- Same one-way trap as file-level KO — applied to tests

Not All Tests Cost the Same

- One build: 35,497 tests · 11.25 h total execution time
- Median test: 0.004 s. Slowest: 213 s. Ratio: 53,000×
- 65% of tests finish in ≤ 10 ms — near-zero cost
- Removing the cheapest 80% saves $< 3\%$ of CI time
- The problem is not too many tests — it's too many *expensive* tests with no justification

Duration Concentration

- Top 1% of tests (317) → 20% of total execution time
- Top 10% (3,168 tests) → 80% of total execution time
- Bottom 80% (25,000+ tests) → 2.4% of time
- Gini = 0.88
- 4 test categories account for 89% of CI time

The Missing Mapping

- 76M test executions / week — 2.5B rows in 8 months
- Compliance: requirement → test → evidence
- CI + code review + issue tracker: raw data for traceability graph
- Partially deployed (Xray); never at full scale.
- Graphs should also carry tests execution cost

Synthesis

Certification adds tests. Nothing removes them. Cost concentrates.

- Compliance demands evidence chains that CI does not produce
- Adding tests satisfies auditors. Removing them requires proof nobody can generate.
- Top 10% of tests consume 80% of CI time — which one are necessary?
- Each new framework = more tests, same missing graph, compounding overhead
- Compliance is seen as a burden from development perspective.

4. Delivery Pressure vs. Quality

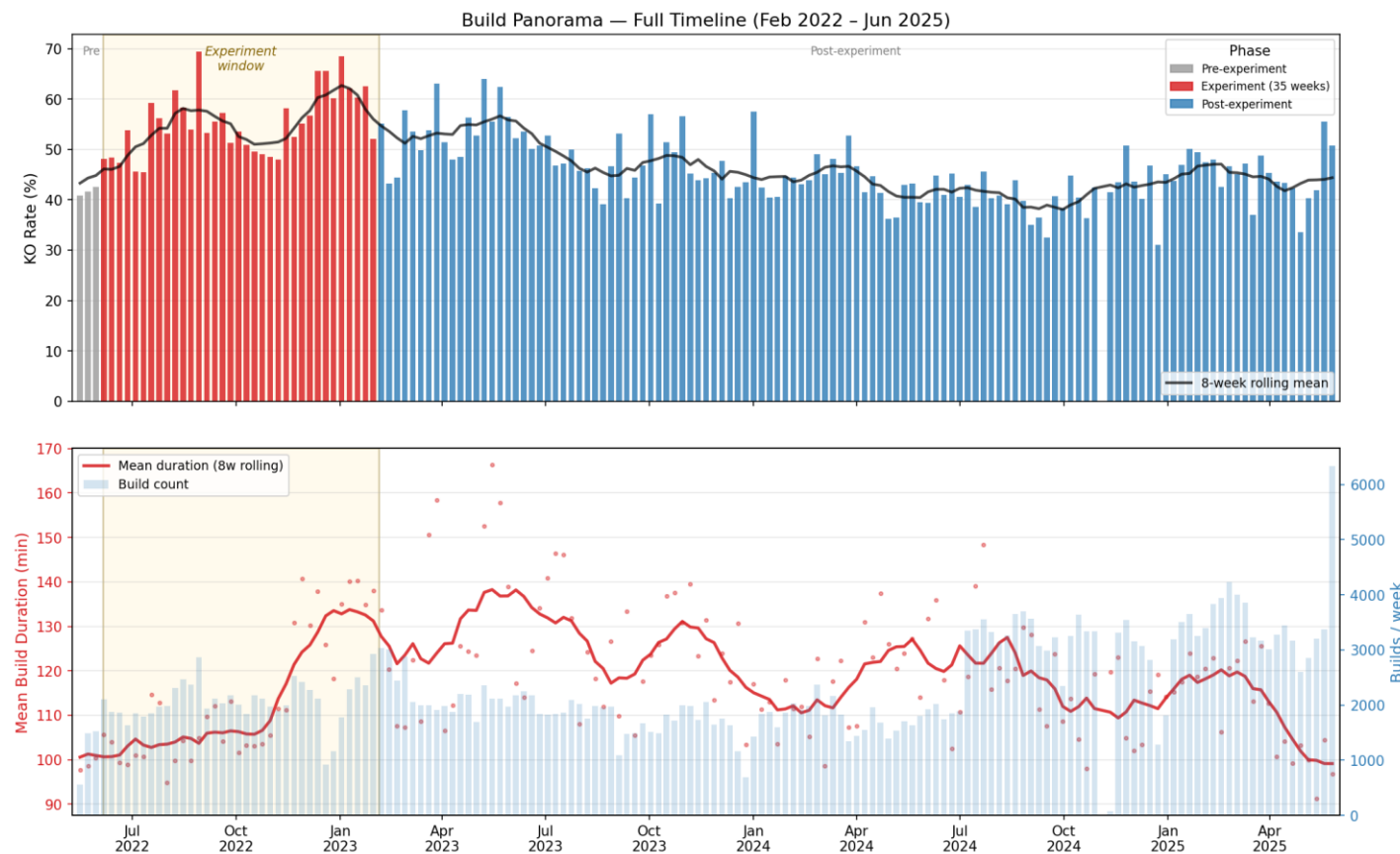


The Business Calendar

- Enterprise revenue concentrated in Q4 — hard delivery deadlines
- Only release-ready features can be announced at annual conference (May)
- Announced in May = sales commitment for Q4 delivery
- Remediation competes with features on the same calendar

The Long View — 3 Years of CI

- Extended dataset: 3 years · 364K builds
- Mean duration: 115 min at start, peaks at 130 min, back to 100 min
- Build KO rate: 48% mean, never below 41%
- Mid-2024: volume doubles overnight



Velocity First, Debt Later

- Feature velocity prioritized for years — observation starts at year 7
- Technical debt accumulated
- Debt is structural: no incremental fix reaches it

The Re-engineering Dilemma

- Fixing architectural debt requires large-scale refactoring
- Refactoring a live system: must guarantee functional equivalence
- Existing test suite fails 48% of the time — can't validate refactoring
- Delivery pressure reclaims the remediation budget — every year
- Rational outcome: live with 48% KO

Synthesis

Velocity-driven debt becomes permanent when safe re-engineering costs more than tolerating failure

- build KO rate 48% over 3 years despite continuous optimization
- Re-engineering blocked: no affordable way to guarantee equivalence
- What is needed: affordable re-engineering with sufficient functional guarantees

5. Tech Shifts: Cloud → LLM



Two Transitions

- First transition: cloud-native + Agile (~10 years)
- Second transition: LLM-assisted development (now)
- Structural issues left unresolved
- The second transition arrives before the first has been fully absorbed

Who Verifies the Agent's Output?

- Agentic AI now writes code, specs, tests, docs, reviews
- Some of these tasks were already done poorly — agents may not be worse
- Verification becomes the limiting factor — and does not scale when done by humans
- Previous sections showed that verification was already failing

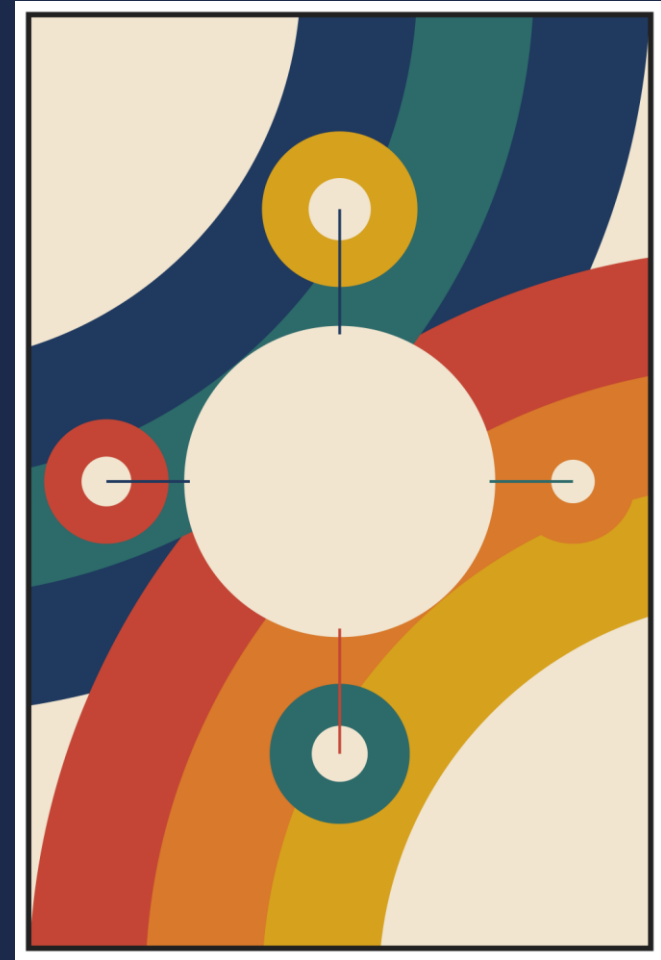
What We Can't See

- Quality beyond pass rates — 0.08% per-test failure rate, yet quality is an issue
- Source code degradation — safe → broken; developers notice, no systemic response
- Which tests for which change? — 38K tests/build, most unrelated to the commit
- Check test necessity — 10% of tests consume 80% of CI time; which expensive tests are dispensable?

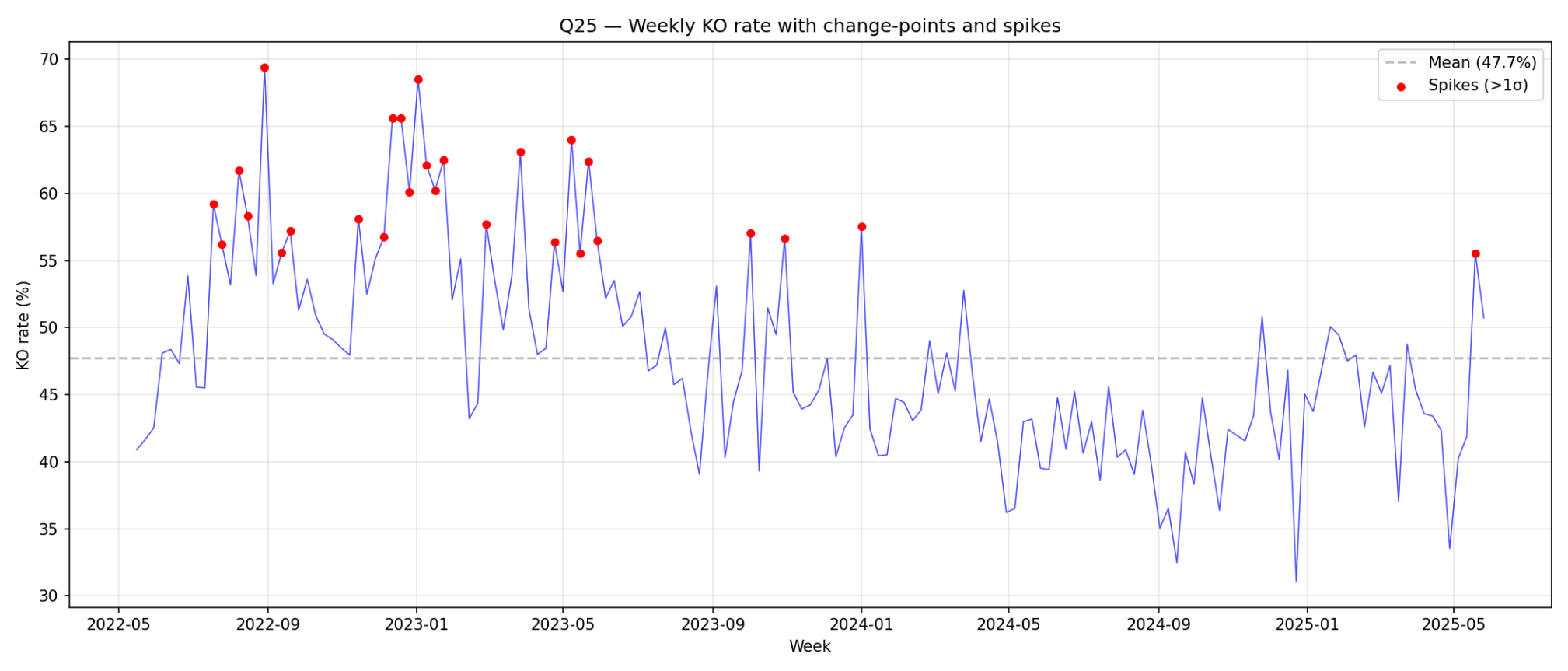
What We Can't Do

- Exploratory test generation — 74% of tests find nothing, experts find bugs in minutes
- Functional equivalence for refactoring — 48% KO suite can't validate rewrites
- Verifying agent-generated artifacts
 - Code correctness against specifications
 - Logical properties on generated specifications
 - Test adequacy to specifications

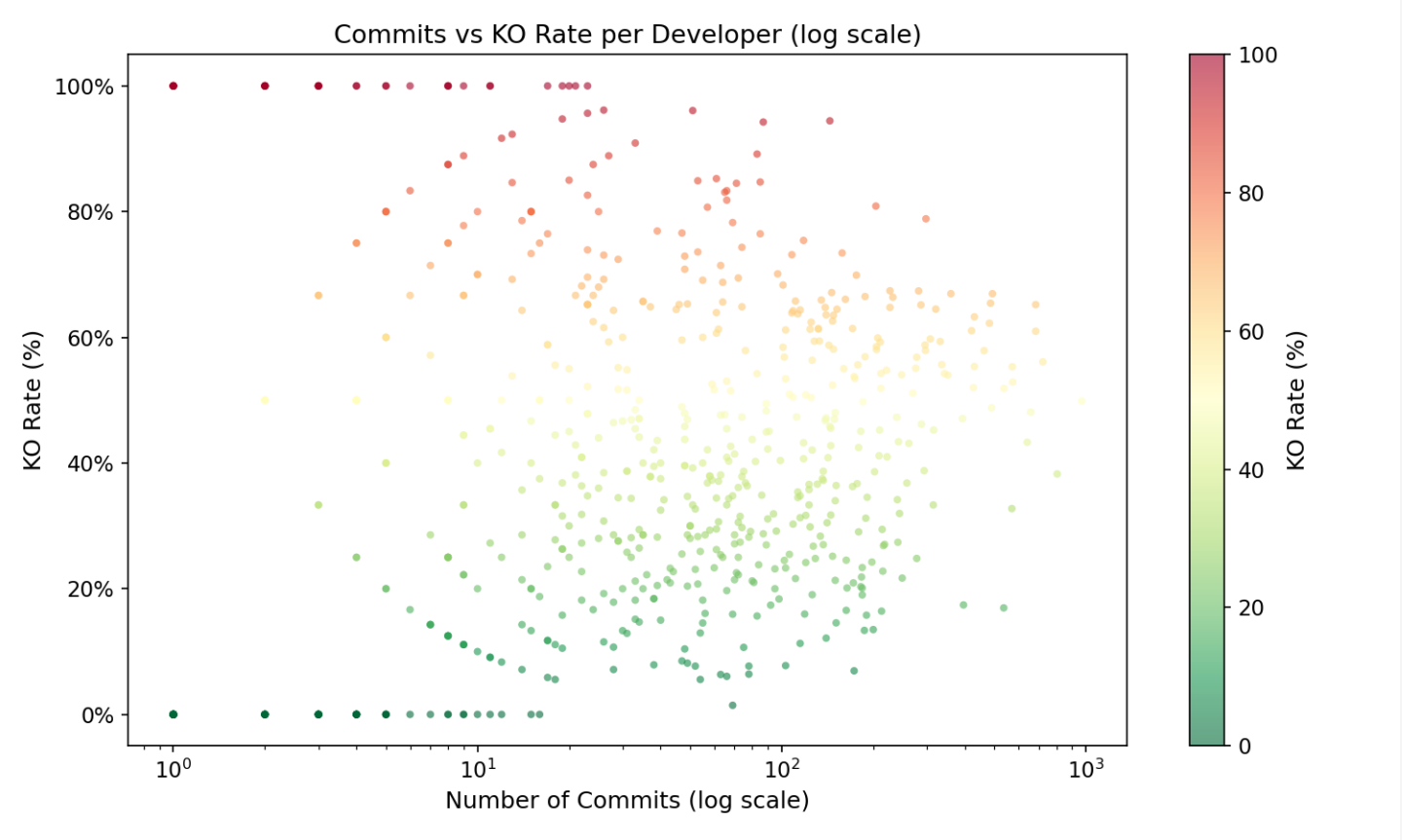
Questions?



RESERVE



RESERVE — KO vs Commits Scatter



Problem — Which Tests for Which Commit?

- Each build runs ~38K tests — most unrelated to the submitted change
- Build time: ~2 hours. Right selection → minutes ?
- Selecting the right tests requires a code → test dependency graph
- 526K test IDs, implicit dependencies — the graph does not exist