July 2026

Measuring the frequency and severity of drift

IN AI-GENERATED AND HUMAN-AUTHORED CODE



REWEAVER.AI
info@reweaver.ai

AI writes more code. It doesn't write production-ready code.

The promise was clear: AI coding tools would accelerate development. They have. Task throughput, epic completion, and PR merge rates are measurably up.

What the velocity charts do not show is what is accumulating underneath: bugs, incidents, rework, and drift from the standards the code was supposed to follow. The industry learned to count output. It never learned to count how far that output had drifted from ready.

WHAT THE DATA SHOWS

+54%
bugs per developer as AI adoption deepens
Faros AI, 2026

96%
of developers don't trust that AI produces functionally correct code
Sonar, 2026

8.5x
code churn (Lines deleted vs. added) per quarter
Faros AI, 2026

4.8x
lead time from code commit to production
Faros AI, 2026

+31%
more PRs merged with no human or agentic review
Faros AI, 2026

1.7x
more issues per PR in AI co-authored code vs . human
CodeRabbit, 2025

Sources: Faros *AI Engineering Report 2026* (22,000 developers, 4,000 teams, 2 years of telemetry); CodeRabbit *State of AI vs. Human Code Generation Report 2025* (470 GitHub PRs, Poisson rate analysis); Sonar *State of Developer Code 2026* (Survey of 1,149 developers)

CONTEXT

Why existing metrics miss the problem

Code quality tools measure static properties: coverage, linting, bug counts. They count what went wrong without accounting for the cost of fixing it.

Invisible problems. A codebase can pass every linter and ship on schedule while carrying months of hidden remediation cost, until the simple feature takes two weeks and nobody can say why.

All drift is not created equal

YOU NEED TO KNOW WHAT IT WILL COST YOU TO FIX.

This study is a controlled comparison of the frequency and cost of drift across eight production readiness dimensions in five leading AI coding tools and a human-authored reference .

5
AI TOOLS
Cursor, Claude Code, Lovable, VS Code, Figma Make

42
PROMPTS
Identical component prompts run across every tool

6
HUMAN REPOS
Repos collected on GitHub and fingerprinted as human-authored

8
DIMENSIONS
Production readiness dimensions measured per file

TWO METRICS

Drift Frequency
Percentage of lines containing at least one drift occurrence. Counts what went wrong.

Production Drift Ratio (PDR)
Drift frequency weighted by remediation cost.

METHODS

Code sampling
Six open-source repositories from a corpus of 21 were fingerprinted and verified by human review as human-authored. AI authored code was derived from identical prompts given to Cursor, Claude Code, Lovable, Figma Make, VS Code + Copilot — all using Claude Sonnet 4.6 as the common model baseline.

Instrumentation
ReWeaver, a deterministic drift-scanning engine. (210 scans, 0 failures. Single detector vintage 2026-07-24), also identifies and calculates PDR.

Procedure
42 identical prompts (buttons, forms, dashboards, navs, modals, auth surfaces) run across all five tools and compared to the human baseline. ReWeaver scanned for drift and calculated frequency and severity.

Metrics
Drift frequency: distinct (file, line) pairs with ≥1 finding ÷ scanned lines.

Production Drift Ratio: frequency weighted by severity × fix-time, normalized per scanned file. PDR 0.30 ≈ 45 min cleanup per scanned file; 0.70 ≈ 2.5 hours. “Low” ≤0.3, “Moderate”=0.3-0.5, “High”=0.5-0.7, and “Severe”>0.7.

Statistics
Two-tailed Wilcoxon signed-rank tests compared AI code (n=5 tools) to human baseline (n=6) globally and on each dimension. Because of the small N, effect sizes and directionality were used to document the strength of the relationship.

Results
The frequency of drift did not meaningfully differ between human- and AI-authored in six out of eight dimensions. **The cost/severity of drift (captured by the PDR) showed dramatic increases (1.5-22x) in AI-generated codebases.**

HEADLINE FINDINGS

The numbers that change how you think about AI code quality.

100%

DRIFT IS ENDEMIC

Both humans and all tools produced drift. No tool won across all dimensions. Instead, tools showed varying profiles of drift.

6 of 8

DIMENSIONS HAD NO CONSISTENT RANK DIFFERENCES IN FREQUENCY

Only Security & Privacy (humans lower drift) and Testability (human higher drift) had differences approaching significance.

6.5×

MORE COSTLY USER EXPERIENCE DRIFT

AI tools also produced 5x more costly Accessibility drift, and 4x more costly Design Consistency drift.

22×

MORE COSTLY SECURITY & PRIVACY DRIFT

AI tools produced 3× the human drift frequency but a cost that was 22× higher. Security drift looks frequency manageable. The PDR reveals it isn't.

STATISTICAL RESULTS

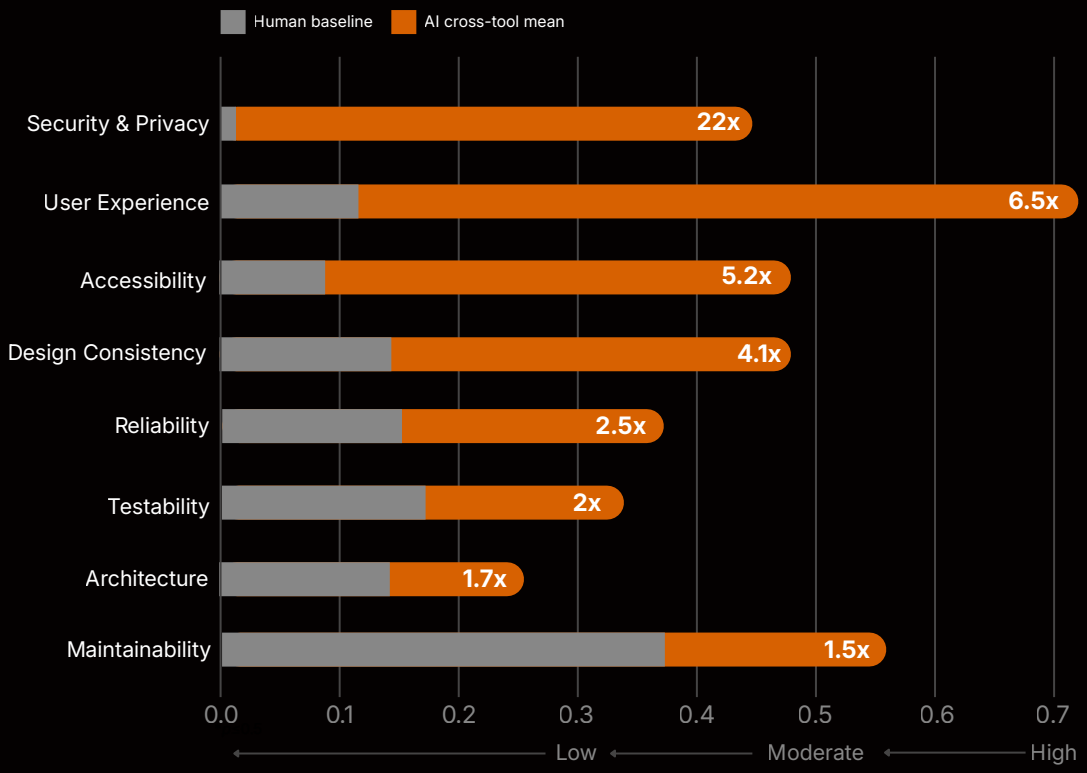
Global assessment of AI- and human-authored code, **AI tools produced significantly more costly drift** ($z = -5.43$, $p < .001$).

In six of eight dimensions, AI tools scored above the human reference with unanimous positive ranks (approaching significance) and large effect sizes ($r = 0.90–0.91$).

Both humans and AI create drift.
But AI creates drift that is more expensive to fix and it happens at a much faster pace.

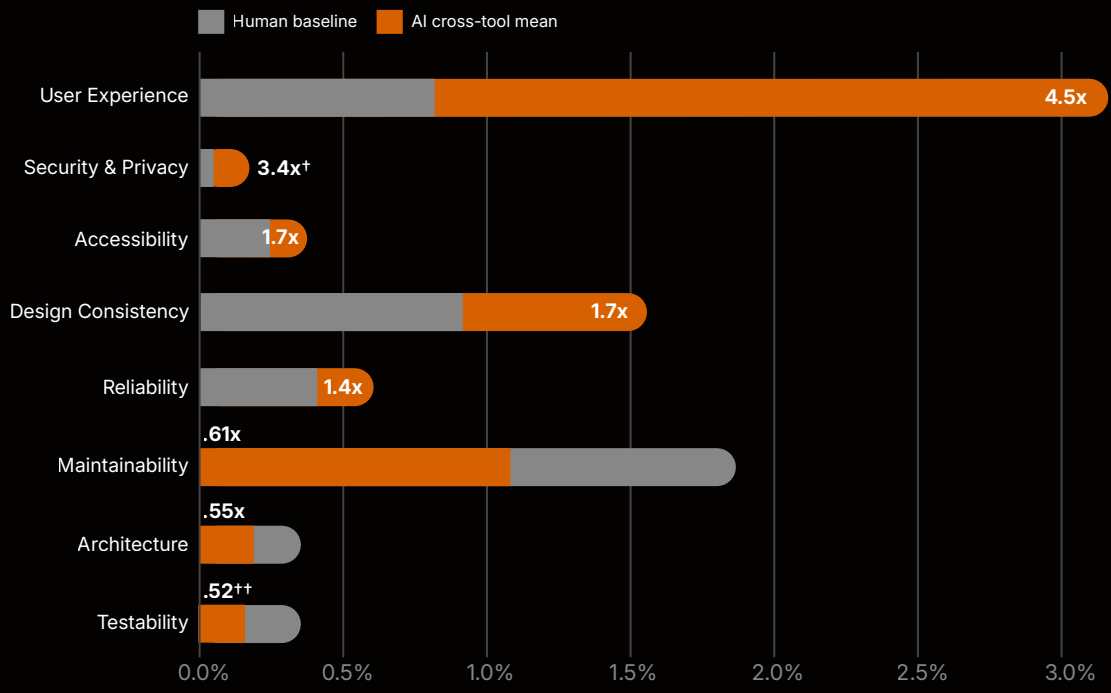
PRODUCTION DRIFT RATIO

PDR scores on a scale of 0 (no drift) to 1 (severe). A PDR of 0.30 means roughly 45 minutes of cleanup per component; 0.70 means about 2.5 hours per component.



DRIFT FREQUENCY

% of lines with ≥1 drift occurrence · AI vs. human-led baseline



† Security & Privacy: All five tools produced higher frequency
†† Testability: All five tools produces lower frequency

WHERE THE PDR CHANGES THE PICTURE

Frequency understates the risk.

AI tools produced three times the human drift frequency in Security & Privacy— a gap that looks manageable until weighted by remediation cost, at which point the PDR is **22× the human reference**. This divergence is not an edge case. It reflects a structural property of AI-generated drift: models produce code that looks syntactically correct and passes review, while remediation costs are concentrated in findings that require non-trivial structural intervention.

How much more drift do AI tools produce?

Dimension	Frequency multiplier	PDR multiplier	
Security & Privacy	3.4×	**22×**	PDR is 6.5× larger than frequency
User Experience	4.5×	**6.5×**	AI created silent failures that would pass tests
Accessibility	1.7x	**5.2×**	Structural remediation required
Design Consistency	1.7×	**4.1×**	Model names from memory
Reliability	1.4×	**2.5×**	Directional, consistent
Testability	0.61×	**2.0×**	Corpus maturity effect
Architecture	0.55×	**1.7×**	Corpus maturity effect
Maintainability	0.52×	**1.5×**	Corpus maturity effect

WHAT DRIFT LOOKS LIKE

Not just malformed code. Absent code.

AI-generated components satisfied the nominal use case while systematically omitting the conditions surrounding it. The drift identified was not due to malformed code — it was **absent code**: missing states, missing boundaries, missing tests. Omissions accumulate silently until a real user encounters the missing empty state, the untrapped focus, or the form that fails without explanation.

SECURITY & PRIVACY

- Role gates enforced only in the UI — bypassable in DevTools.
- Raw PII and credentials passed through props without tokenization.
- The model treated the browser as trusted.

ACCESSIBILITY

- Focus escaped modals and was never returned, leaving keyboard and screen-reader users disoriented.
- Interactive elements built without semantic markup.

USER EXPERIENCE

- Containers missing overflow containment.
- Silent form failures: errors that name the problem but not the fix, empty lists with no empty state, constraint hints before interaction.

DESIGN CONSISTENCY

- Models named UI primitives from memory without verifying they existed in the design system.
- Inline styles bypassed design tokens.
- Placeholder content shipped live.

MAINTAINABILITY

- Hardcoded English strings instead of i18n keys.
- Demo data as production strings.
- Hallucinated HTML attributes browsers ignore.

RELIABILITY

- Missing memo-stable references: inline objects creating a new reference every render, invalidating memoization, and retriggering effects silently.

ARCHITECTURE

- Code six-plus levels deep.
- UI reaching directly into services.
- Top-level mutable state leaking across importers.
- Render logic too branchy to test.

TESTABILITY

- Document/window reached synchronously in component bodies — unmockable.
- Components fusing fetch, DOM, and business logic.
- The silent killer: no tests emitted alongside components.

MAIN CONCLUSIONS

Drift happens. Knowing the cost is key.

Every tool tested — regardless of benchmark performance or commercial positioning — produced meaningful drift across all dimensions.

Tool selection does not substitute for post-generation verification. A team who switches tools hoping to reduce drift will simply change its drift profile, not eliminate the drift.

Raw frequency, considered alone, substantially understates the risk of drift.

In the Security & Privacy dimension, AI tools produced three times the human drift frequency, a gap that looks manageable until weighted by remediation cost, which was 22x the human reference.

The divergence between frequency and severity reflects a structural property of AI-generated drift.

Models produce drift that looks correct and passes review, while remediation costs are concentrated in findings that require non-trivial structural intervention. Frequency counts what went wrong. The PDR quantifies what it will cost to fix it.

The actionable conclusion is not which tool to use; it is that any tool requires a verification layer capable of catching what generation leaves behind.

REFERENCES

Becker, J., Rush, N., Cunningham, T., Rein, D., & Mahamud, K. (2026). *We are changing our developer productivity experiment design.* METR.

CodeRabbit. (2025). *State of AI vs. human code generation report.*

Cortex. (2026). *Engineering in the age of AI 2026: Benchmark report.*

Cui, K. Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2025). The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers. *Management Science.*

Faros AI. (2026). The AI Engineering Report: The acceleration whiplash.

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot. *arXiv*:2302.06590.

Sabra, A., Schmitt, O., & Tyler, J. (2025). Assessing the quality and security of AI-generated code. arXiv:2508.14727.

Vaddepalli, R. K. (2026). Monitoring the unseen dependency: A framework for detecting LLM drift-induced failures. *Computing & Data Science Innovations, 1*(1).