■ BENCHMARK REPORT

Autonomous unit test generation at enterprise scale

Diffblue Testing Agent vs. Claude Code

2.5x	8%	~200x	37%
more code covered (81% vs 32%)	better test strength (82% vs 74%)	more lines covered per minute of dev time	more mutations caught (61% vs 24%)

diffblue.

DIFFBLUE.COM

Executive Summary

The challenge:

AI code generation tools have been adopted by software development teams at an astonishing pace over the past two years. And while many of their capabilities often seem magical, they still struggle mightily in some use cases, particularly those that require complex, long-running workflows that require near-perfect output to be effective. Automated regression unit test generation for large code bases is a prime example of this problem, and it is a problem faced by most companies – particularly those that have poorly covered legacy applications in need of modernization. This is the precise use case for which the Diffblue Testing Agent was built.

The experiment:

The purpose of this benchmark study is to quantify the differences between two distinct approaches to generating unit test coverage at project-level scale: a senior developer armed with a state-of-the-art coding agent (Anthropic's Claude Code using the latest Sonnet and Opus 4.6 models) vs. the autonomous Diffblue Testing Agent, a new product designed to orchestrate any coding agent's behavior to deliver maximum test coverage without developer intervention.

To accomplish this, we selected 8 Java repos that are representative of the kinds of codebases enterprises are trying to modernize today. We ran the Diffblue Testing Agent against these repos and collected key performance metrics, including total line coverage, test strength, and mutation coverage.
We then challenged an experienced, AI-savvy, senior Java developer to see how much coverage they could generate in 2 hours of their time or with a maximum of 20 prompts using Claude Code.

The results:

The study's findings suggest that it may be infeasible – or at least impractical – to rely solely on AI coding agents to achieve comprehensive test coverage. Within the constraints of our experiment (2 hours or 20 prompts), we **achieved an average line coverage** of only 36% with Claude. Even more troubling, we saw rapidly diminishing incremental returns as the investment of time and energy continued, indicating that a ceiling of practical potential coverage may be well below 50% – an outcome that is far below the 80%+ target coverage that most development organizations mandate. Additionally, the experience of using Claude to attempt to generate broad test coverage was one that required constant developer engagement to monitor the agent's performance and remain available for frequent interactions at unpredictable intervals. In many ways, this is the form of developer toil in the AI era: the need for constant agent babysitting where constant context switching renders the much hoped-for productivity gains elusive.

On the other hand, the Diffblue Testing Agent (DTA) delivered far higher test coverage – 81% on average and more than twice that achieved by Claude – with no continuous developer supervision required. The agent ran autonomously – sometimes for several hours – delegating method- and class-level test creation to Claude while orchestrating a comprehensive process that includes coverage analysis, build system fixes, test plan creation, parallelized test generation, output verification, project clean-up, and PR preparation. The resulting tests were, on average, of higher strength than those generated by the developer using Claude (82% vs. 73%), and the mutation coverage on the DTA-generated test suite was 2.5x times better, primarily due to the higher overall line coverage achieved.

Request an evaluation

Bottom line:

The Diffblue Testing Agent delivers the comprehensive test coverage required to modernize legacy applications safely, while AI coding tools in the hands of expert developers fall far short of the target. For all their impressive capabilities, tools like Claude Code and GitHub Copilot still struggle with complex, long-running workflows. Requiring developers to use them for tasks they are ill-suited for creates a new form of AI-era toil that kills productivity and job satisfaction.

Key findings:

METRIC	DIFFBLUE TESTING AGENT	DEVELOPER + CLAUDE CODE	DELTA
Avg. line coverage	80.7%	32.3%	2.5x more coverage
Avg. mutation coverage	61.3%	24.2%	2.5x more mutations killed
Avg. test strength	81.8%	73.9%	8% stronger tests
Developer Experience	Single prompt	Continuous 'agent-sitting'	Genuine automation
Productivity	3,884 lines per prompt	67 lines per prompt	58x more productive

† Diffblue Testing Agent requires minimal developer time for initial configuration and launch (estimated 30sec per repository). All test generation, compilation, and verification is fully autonomous.

Contents

Methodology

Repositories tested

As the foundation of our experiment, we selected 8 real-world Java projects to test. Some were open-source projects and others were closed-source code production code from large enterprises. We've anonymized the project names in the table below to respect the confidentiality of closed-source projects. While some projects included modest levels of pre-existing test coverage, we deleted all tests before starting to use the two competing approaches to generate test coverage.

REPOSITORY	DOMAIN	COVERABLE LINES
Repo 1	Enterprise services	3,586
Repo 2	Messaging platform	3,517
Repo 3	Metadata management	4,323
Repo 4	JSON serialization	5,021
Repo 5	Storage / OSGi	8,712
Repo 6	Financial trading	2,521
Repo 7	Data compaction	1,050
Repo 8	Derivatives / DeFi	2,339
TOTAL		**31,069**

Test setup

Diffblue Testing Agent was run in batch mode against each repository. A developer performed initial configuration and launch (generously estimated at 1 minute per project), after which the agent operated fully autonomously: ingesting the codebase, generating tests, compiling, running, and verifying — discarding any test that does not compile or pass. Depending on the size and complexity of the project, the agent took anywhere from 15 minutes to 5.5 hours to complete.

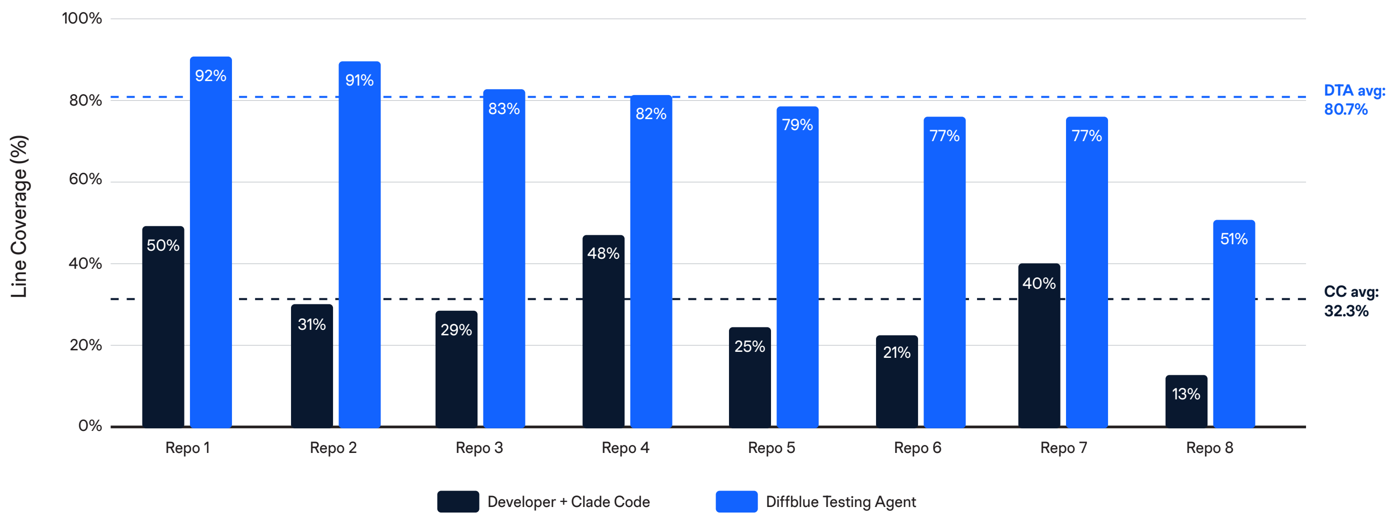
Developer + Claude Code was operated by senior Java engineers (multiple team members, ~10 years average experience, deep knowledge of testing, and daily users of AI coding tools) who attempted to use Claude Code to generate tests for the entire project, reviewed output, and iterated on prompts until they hit a 2-hour or 20-prompt limit.

Metrics measured

Line coverage — Percentage of coverable lines executed by the generated test suite.
Mutation coverage — Percentage of code mutations (seeded bugs) killed by tests, i.e. assertions fail. Proxy for defect detection power.
Test strength — Percentage of covered mutations that are killed. Measures assertion quality independent of coverage.
Human effort — Developer minutes invested, number of prompts, manual fix cycles.

Results: Line coverage comparison

Diffblue Testing Agent achieved consistently higher line coverage across all 8 repositories, averaging 80.7% compared to 32.3% for Developer + Claude Code — a 2.5x advantage.



Why the gap?

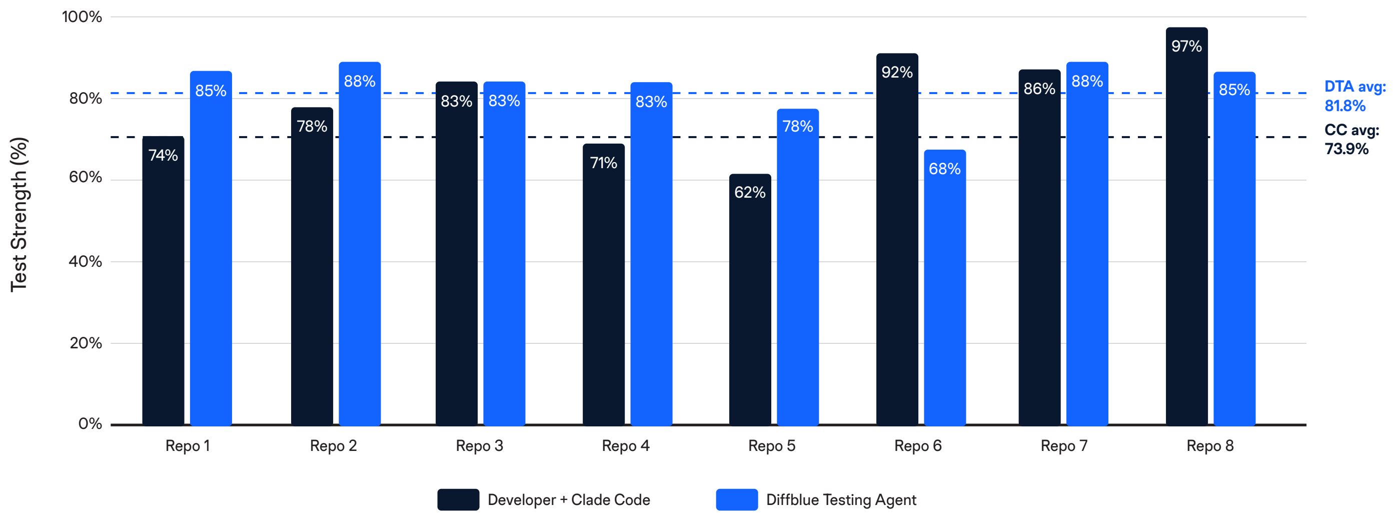
Autonomous codebase-scale operation: Agent supervision is the hidden cost: Claude Code doesn't just need prompts — it needs a human keeping it on track. In our benchmark, the agent repeatedly went off-plan: skipping modules it had committed to cover, stopping before completing its own task list, and claiming work was done that hadn't been. The developer's primary job was not writing tests but supervising the agent. Diffblue Testing Agent performs this supervisory role autonomously — managing the generation plan, verifying actual completion against the intended scope, and ensuring nothing is missed. That's the difference between an agent that assists with code generation as opposed to an agent that delivers and orchestrates and validates a complex, long-running process.

No diminishing returns: Developer fatigue, context-switching, and time constraints naturally limit the Claude Code approach. Diffblue Testing Agent maintains consistent throughput regardless of codebase size or complexity.

Results: Test quality

Test strength — comparable per-test quality

Test strength measures the quality of individual test assertions: of the mutations covered by each test, what percentage are actually killed? Here, the Diffblue Testing Agent delivered modestly but meaningfully better results, averaging test strength scores of 81.8% vs 73.9% for Developer + Claude Code (1.1x). Given that Claude was the test generation engine in both scenarios, the better scores delivered by the Diffblue Testing Agent speaks to value added by an agent orchestrating an opinionated workflow that includes a test review and verification step.

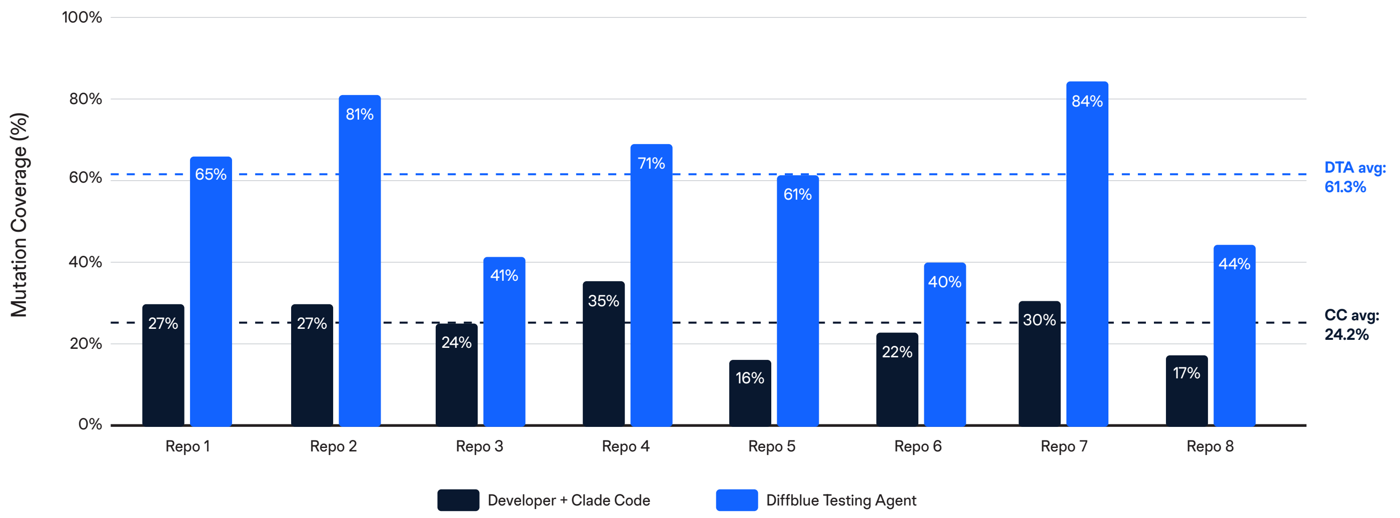


Key insight:

Diffblue delivers stronger tests than a developer using Claude.
This advantage is compounded by the fact that Diffblue also generates far higher test coverage.

Mutation detection — high test quality and broad coverage drive better defect detection

Mutation coverage measures the total defect-detection power of the test suite. Because Diffblue Testing Agent covers 2.5x more code, it also kills 2.5x more mutations: 61.3% vs 24.2%. This is the metric that matters for regression safety before a modernisation programme.

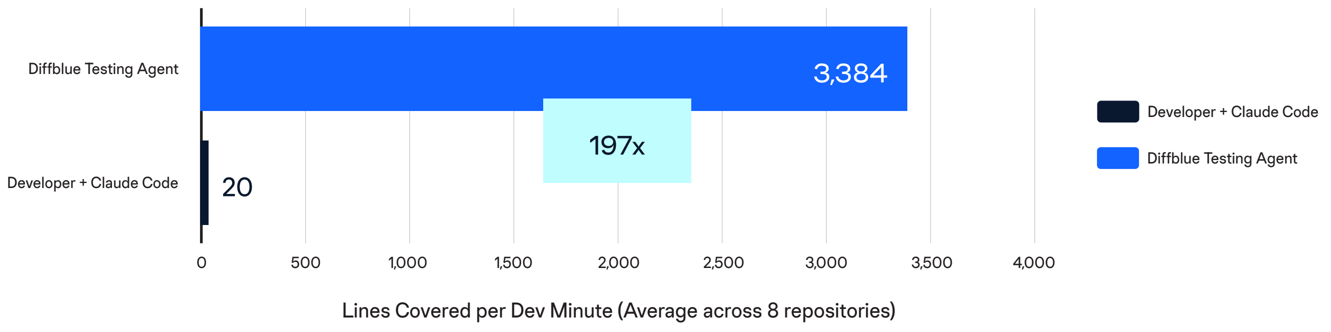


Results: Productivity

Productivity in test generation comes down to a simple question: how much verified coverage does each minute of developer time produce?

197x more lines covered per developer minute

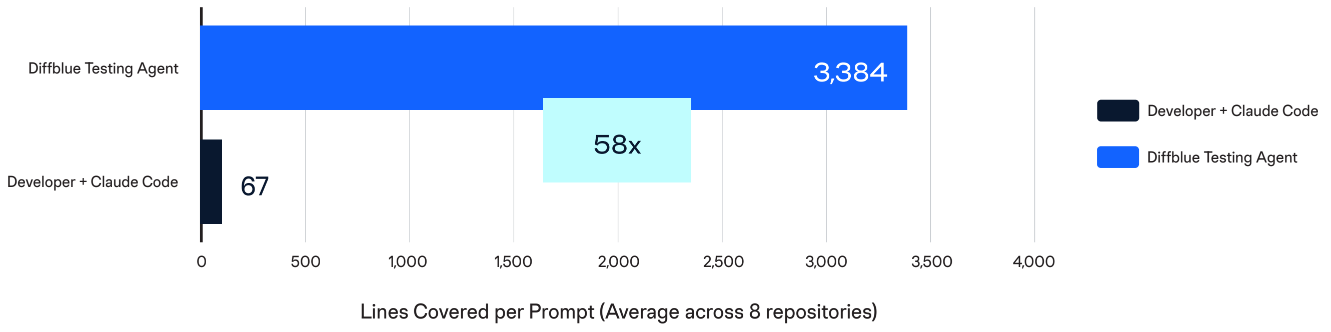
With Developer + Claude Code, a senior developer invested an average of 64 minutes per repository — actively prompting, supervising, and correcting the agent — and produced an average of 20 lines of coverage per minute of their time. Diffblue Testing Agent, after minimal initial setup, produced 3,384 lines of coverage per developer minute invested: **a 197x advantage.**



DTA runs autonomously for hours while the developer works on something else. Developer + Claude Code requires continuous, hands-on attention for the duration of the session, and as our benchmark team observed, much of that attention is spent not on productive test authoring but on supervising an agent that drifts off-plan, claims false completions, and abandons tasks mid-execution.

58x more lines covered per prompt

Each prompt in the Developer + Claude Code workflow represents a developer decision: what to target, how to instruct the agent, and how to recover when the output isn't usable. Across 8 repositories and 149 prompts, each prompt yielded an average of 67 lines of coverage. DTA's single-prompt-equivalent setup yielded 3,384 lines per prompt: **a 58x advantage.**



The prompt-level metric matters because it captures the cognitive overhead of the manual approach. Every prompt requires the developer to context-switch, evaluate the agent's prior output, decide whether to iterate or move on, and formulate a new instruction.

What this means for planning

If your modernisation target requires regression coverage across 100,000+ coverable lines, the manual approach at the rates observed in this benchmark would require roughly 5,000 developer minutes (~83 hours, or more than two working weeks of a senior engineer's undivided attention) and ~1,500 prompts and would still likely plateau well below the 80% coverage threshold. DTA delivers that coverage autonomously.

Conclusions

The difference between Diffblue Testing Agent and Developer + Claude Code is scope, autonomy, and what the approach demands of your most expensive resource: senior developer time.

2.5x more regression coverage

Diffblue Testing Agent covered 80.7% of lines across 8 repositories vs 32.3% for Developer + Claude Code. The study also showed rapidly diminishing returns of continuing to invest developer time in coaxing more coverage out of Claude, suggesting that this coverage gap cannot be closed.

2.5x more defect detection power

Mutation coverage — the proxy for real-world bug detection — was 61.3% vs 24.2%. More code covered means more mutations caught.

Superior test quality

Test strength was 81.8% vs 73.9% (1.1x). Both approaches generate solid tests, but those generated by the Diffblue Testing Agent are stronger.

No “agent-sitting” required

The developer experience of using Claude to generate regression test suites is one of constant supervision and interaction at unpredictable intervals. The constant context switching that results is a silent productivity killer. In contrast, the Diffblue Testing Agent runs autonomously until the job is done, freeing up developer time to focus on other things.

The real risk is the coverage gap

The coverable lines left untested by the Claude Code approach represent unprotected code during refactoring, upgrades, and modernisation. That gap is the business risk this benchmark quantifies.

Ready to see these results on your codebase?

Diffblue Testing Agent runs against your real repositories — Java and Python — and you only pay for verified, passing tests.

🔗 Request an evaluation

diffblue.

DIFFBLUE.COM