

Stop reviewing the same engineering decision twice

Decreen makes approved engineering decisions reusable across AI-assisted development.

AI can generate a pull request in minutes. A senior engineer still has to answer: does this change conflict with something our team already decided?

A decision might be: "The Orders service owns order data." The rule that follows: "Other services must not write directly to the orders database."

These decisions live in code, tests, documentation, review comments, and senior engineers' memory. As agent-generated changes increase, teams keep having to reconstruct decisions they have already made.

How Decreen works

For each change, Decreen automatically shows:

- which approved decisions apply,
- what evidence supports them,
- whether the change conforms or conflicts,
- and where a genuinely new decision is required.

Evidence so far

Decision discovery

On a second small repository outside Decreen's own codebase, Decreen surfaced 30 reviewable engineering decisions from existing repository knowledge. They were human-reviewed before becoming authoritative.

We also exercised an independent code-first discovery path. A repository mechanism scan found 100 concrete mechanisms — tests, guards, schema constraints and similar structures — and produced 80 Candidate LAW proposals from them.

These are complementary discovery paths: one starts from what the team has written down, the other from what the repository appears to constrain.

We have not yet measured discovery recall — how many of the repository’s true engineering decisions these methods find — or the final candidate noise rate across multiple repositories.

Decision enforcement

Once a decision is approved, Decreen asks a stricter question:

What, if anything, mechanically makes this exact decision true?

In a manually calibrated set of five decisions known to have enforcement, Decreen found the relevant enforcement location in all five cases.

The harder problem is determining how much of the decision that mechanism actually guarantees. A mechanism may cover only part of a rule, leave a bypass path, or merely reflect current behavior rather than prevent a violation.

Discovery evidence can help locate relevant code, but it is not itself proof of mechanical enforcement.

Judge

On one real code change, Decreen evaluated 20 relevant **decision interactions**:

- 16 conformed to existing architectural intent,
- 2 required new engineering decisions,
- 1 appeared to violate an existing decision,
- 1 exposed a gap in the architectural model.

The apparent violation was particularly useful. The code itself was not the problem: the existing rule had been written too broadly. The appropriate response was to narrow the rule rather than revert the change.

That is the distinction Decreen is intended to make: is this a wrong change, an imprecise existing decision, or a new decision that requires human judgment?

We then repeated the core Judge test on the second repository, with the authoritative decisions frozen before the tested changes. Across three controlled cases — a conforming change, a violation, and a genuinely new decision — Decreen identified the correct class in all three.

These are encouraging controlled results on a limited number of cases and repositories. They are not yet a claim of general discovery recall, enforcement accuracy, or Judge accuracy across arbitrary codebases.

Where Decreen runs

Decreen runs in the customer-controlled execution environment: for example a local agent runtime, developer workstation, self-hosted CI runner, or GitHub Actions runner.

The Decreen CLI and optional skills operate against the repository there. The repository itself does not need to be uploaded to Decreen. Decreen can synchronize derived product state such as approved decisions, references, workflow state, and review results.

Data handling by the customer's chosen agent runtime, CI provider, or model provider remains subject to that provider's own configuration and policies.

Proposed pilot

Pilot scope: one engineering area with enough activity to review 30–50 historical PRs and observe 20–30 new changes.

The scope can be a service, subsystem, product area, or a clearly defined slice of a monorepo. We choose it together so it is small enough to have coherent

engineering decisions, but active enough to produce a useful sample of changes.

The pilot is defined by the number of changes, not by a fixed calendar duration. A high-throughput team might complete it in a few weeks; a lower-throughput team may take a few months.

Phase 1 — Backtest & decision baseline

Fixed price: \$2,500

- Review 30–50 historical PRs in the agreed engineering area.
- Surface candidate engineering decisions from existing repository knowledge.
- Senior engineers review and approve the 15–25 decisions that matter most.
- Verify what evidence and enforcement mechanisms support those approved decisions.
- Deliver a written backtest report: the decisions surfaced, which past changes interacted with them, and which decisions appear weakly enforced or unenforced.

Phase 1 answers one question: is the signal useful enough to take Decreen live?

Expected duration

Usually 1–2 weeks once access, scope, and the PR sample are agreed. The calendar time depends mainly on how quickly the review sessions can be scheduled, not on repository size alone.

Phase 2 — Live validation

Fixed price: \$2,500

- Run Decreen on the next 20–30 relevant changes in the same engineering area.
- Check those changes against the approved decision baseline.
- Track conforming changes, violations, genuinely new decisions, and model gaps.
- Review whether Decreen reduces repeated senior review and helps isolate where new judgment is actually required.
- Remain flag-only: Decreen does not silently approve or block merges.

Phase 2 ends when we have observed 20–30 qualifying changes, not after a fixed number of weeks.

Success means:

- fewer repeated architectural review comments,
- faster identification of genuinely new engineering decisions,
- clearer visibility into decisions that are weakly enforced or unenforced,
- less senior-review time spent re-establishing context.

How we choose the engineering area

A good pilot area has:

- a recognizable owner or senior reviewer,
- a coherent set of engineering decisions,
- enough recent PR history for a 30–50 PR backtest,
- enough ongoing activity to produce 20–30 new changes in a reasonable time.

For a monorepo, the pilot does not need to cover the whole repository. We can scope it to a service, package group, domain, or subsystem with clear boundaries.

What it takes from your team

- One engineer: about half a day to install Decreen and run the first analysis.
- Senior engineers: two 1-hour sessions to review and approve the 15–25 decisions that matter most.
- During Phase 2: short review of the changes that Decreen flags.

Pilot terms

- Phase 1 — \$2,500 fixed price.
- Phase 2 — \$2,500 fixed price.
- You decide whether to continue after Phase 1.
- The pilot duration depends on engineering throughput, not a fixed contract length.

Next step

A 30-minute call to pick the repository and agree on what success looks like for your team. We can start the backtest within a week of that call.

At the end of month 1 you get a written backtest report. It is yours to keep whether or not you continue.

The question

Would you run Decreen on one repository and its past PRs to test whether it reduces repeated senior review in your AI-assisted development workflow?

[Book a 30-minute call](#) →

[Try a free system map](#) →

alina@decreen.com