SEVEN PRODUCTS. ONE SHARED BRAIN. BUILT FOR SOFTWARE'S NEW LIFECYCLE: THE ADLC.

Govern the code your AI writes. Prove *the ROI.*

Garth doesn't sit beside your stack. It works inside it: one brain governing your code, scanning your repos, assisting your developers where they already are, and giving leaders the signal to actually lead.

IN THIS DOCUMENT

THE SUITE

Every AI product sold to enterprise leaders makes the same promise: *we'll make your people faster.* A fine promise, and an incomplete one. AI now runs through every stage of how software gets built: it suggests, it reviews, it secures, it ships, it gets measured. That is the AI-assisted development lifecycle (ADLC), and most tools sharpen a single stage of it while the seams between them quietly widen. Faster people working on the wrong things, reviewing code without grasping the architecture it lives in, hunting for knowledge siloed three tools away. That is not a productivity win. It is a more expensive version of the same problem.

Garth was built for the question nobody asks in the demo: **what happens the next morning?** When the sprint resets. When two engineers are unknowingly modifying the same module on parallel branches. When the new engineer joins and asks where to find the architecture decision from six months ago. When the CTO needs to know which teams are actually getting value from AI, and at what cost per line shipped. Garth was designed for those moments: not as a chatbot you invoke, but as infrastructure that hums.

> *"The question isn't whether AI can accelerate your development process. It's whether your AI knows your codebase, your team, and your organization well enough to be trusted with them."*
>
> The Garth Design Principle

Seven products. One knowledge backbone. Four leadership perspectives, each with a different set of questions, each with a frank answer. We've let them argue, had the PM stitch it, and let the UX tell the final story. Two stories, actually: one for the developer living in the work, one for the leader overseeing it.

Seven products that share a brain.

Each solves a distinct problem on its own, so you can start with one and add the rest as it earns its place. GKS is the invisible layer that makes them coherent, and more valuable together than any one alone.

GREVIEW

AI CODE GOVERNANCE REVIEW

Reviews code against your full repository graph, team guardrails, and inter-PR collision risks, not just the diff in front of it

FOR DEVELOPERS + ENGINEERING LEADS

GSCAN

REPO SECURITY & COMPLIANCE INTELLIGENCE

Continuous repo-level scanning that surfaces issues to developers before they become downstream risks or incident reports

FOR SECURITY + ENGINEERING

GASSIST

AGENTIC DEVELOPER ASSISTANT

AI assistance in Slack and Teams: answers technical questions, resolves blockers, surfaces org knowledge exactly where developers already are

FOR DEVELOPERS, IN THEIR CHANNEL

G-IDE

IDE EXTENSION

GKS-grounded org context layered into VS Code, Cursor, Windsurf, and Antigravity: team-pattern-aware suggestions without leaving the editor

FOR DEVELOPERS, IN THEIR EDITOR

G360

ENGINEERING INTELLIGENCE PLATFORM

AI adoption, token spend vs. lines shipped, tool ROI, and optimization signals: the view leaders need to actually manage the AI era

FOR CTOS, CIOS, CFOS

GRELEASE

RELEASE INTELLIGENCE · IN PREVIEW

Joins Jira commitment, Git landing, CI verification, and cross-team dependency state into one graded release-readiness verdict, with every slipped date on record

FOR CTOS, DELIVERY LEADERS + PE OPERATORS

GARTH UNIVERSE

CENTRAL CONTROL PLANE

Universal configuration, tenant management, licensing, access control, integrations, and module enablement across the entire suite

THE PORTAL FOR ALL OF IT

INVISIBLE BACKBONE · POWERS EVERY PRODUCT ABOVE

GKS · Garth Knowledge System

GKS stores your organizational knowledge as a traversable graph with semantic embeddings: what relates to what, who owns it, what changed, what's stale. Every product in the suite draws on this shared context. Not a search bar. Not a chatbot. The memory that makes Garth coherent.

KNOWLEDGE GRAPH · SEMANTIC EMBEDDINGS · HYBRID RETRIEVAL · STALENESS DETECTION · ORG-SCOPED · MCP-COMPATIBLE · AGENT-CALLABLE · CLI ↗ COMING SOON

What each one actually does.

Architecture, capabilities, and the numbers that matter, per product, per audience. No marketing claims without the mechanism underneath.

▽ FOUNDATION LAYER
POWERS EVERY PRODUCT ABOVE

GKS · Garth Knowledge System

The brain the rest of the suite thinks with.

Every other product in this document is, on its own, a generically capable AI tool. What makes them Garth, *what makes them know your auth flow, your protected paths, the ADR that explains why a module looks the way it does, is GKS. It holds your organization as a traversable graph: what relates to what, who owns it, what changed, what has gone stale.*

The AI-assisted development lifecycle runs on context. Strip the context away and you are left with autocomplete in a confident voice. GKS is *that context, written once, drawn on by GReview when it reviews, by GAssist when it answers, by G-IDE when it suggests. Not a search bar. Not a chatbot. The memory that makes the suite coherent.*

KNOWLEDGE GRAPH | SEMANTIC EMBEDDINGS | HYBRID RETRIEVAL | STALENESS DETECTION | ORG-SCOPED | MCP-COMPATIBLE | AGENT-CALLABLE | CLI ↗ COMING SOON

WHY IT'S THE BACKBONE, NOT A PRODUCT

Remove GReview and you lose governance review. Remove GKS and the other five stop being Garth.

01

GREVIEW

AI CODE GOVERNANCE REVIEW

Reviews the codebase. Then reviews the PR.

Most AI review tools read a diff. GReview reads that diff *in context*, against a live snapshot of your repository's symbol graph, call trees, import chains, and type definitions. It knows the function you just changed is called by fourteen other modules. It knows your team's guardrails. It knows what two other open PRs are also touching this week. That's not a reviewer. That's an architect who never sleeps.

Supports **GitHub, GitLab, Bitbucket, and Azure DevOps (ADO)**. Configurable per-repo via `.reviewconfig.yml`. Async review via RabbitMQ. Your pipeline doesn't wait. OAuth user-token flow means approvals carry real engineer identity.

WHAT MAKES THE REVIEW DIFFERENT

AST precision, GKS memory

Most AI reviewers read the diff, or at best index the repo. GReview reads the abstract syntax tree, so it judges the change by structure rather than text, and it draws on GKS, so it already knows your conventions, the decision record behind the module, who owns it, and what changed last quarter. Structure and organizational memory, in a single pass. It reviews the way a senior engineer who has been at your company for years would.

DIFFERENTIATOR

Inter-PR Impact Surfacing

GReview doesn't just review the PR in front of it. It watches across all concurrent open PRs, surfacing collision risks when two engineers are unknowingly modifying the same module on parallel branches. The kind of integration conflict that normally surfaces on merge day, caught before anyone writes a single conflict marker.

AST-INFORMED REVIEW | GKS-GROUNDED | SYMBOL GRAPH CONTEXT | INTER-PR COLLISION DETECTION | RISK TIERING H/M/L | REPO-SPECIFIC RULE ADHERENCE | TEAM GUARDRAIL ENFORCEMENT | HIDDEN AGENTIC RISK DETECTION | GITHUB · GITLAB · BITBUCKET · ADO | ASYNC · RABBITMQ | PER-REPO .REVIEWCONFIG.YML | OAUTH USER ATTRIBUTION | PROMPT VALUATION ↗ COMING SOON | AUTOMATED TEST GENERATION ↗ COMING SOON

COMING SOON

Prompt Valuation

As AI-generated code enters codebases at scale, the quality of the prompt behind it matters as much as the code itself. Prompt Valuation assesses whether the prompts driving AI-generated contributions are producing quality, safe, auditable output, closing the loop between intent and result.

COMING SOON

Automated Test Generation

GReview will generate and run targeted tests for each PR in a sandbox, catching the edge cases that slip past human reviewers and static analysis alike. Review that does not just read the change, but proves it.

REVIEW TIME REDUCTION
62%
↑ vs. manual-only workflows

REGRESSIONS CAUGHT PRE-MERGE
3.1×
↑ vs. diff-only AI tools

FALSE POSITIVE RATE
4.2%
↓ vs. 18.7% with diff-only tools
Symbol-aware + guardrail-aware review eliminates the noise that erodes engineer trust

REVIEW COMMENTS ACTED ON
90%+
↑ Accepted, not dismissed
Engineers act on almost every comment because almost every comment is relevant. Low false positives are what earn that: GReview stops being another tool you learn to ignore.

02

GSCAN

REPO SECURITY & COMPLIANCE INTELLIGENCE

Finds it before it ships.

Security issues found in production cost an order of magnitude more than issues found in development. GScan runs continuously at the repository level, not as a post-commit afterthought, but as a persistent awareness layer that surfaces issues to developers before they become sprint blockers or incident reports.

Hosted in Garth Universe. Findings are context-enriched: not just "vulnerability in line 42" but why it matters, what it affects, its CVSS score, and what to do next. Configurable rulesets mean different teams can enforce different compliance postures on the same platform.

CONTINUOUS SCANNING | REPO-LEVEL COVERAGE | CONFIGURABLE RULESETS | CONTEXT-ENRICHED FINDINGS | CVSS SCORING | COMPLIANCE REPORTING | HOSTED IN GARTH UNIVERSE

ISSUES CAUGHT PRE-PRODUCTION
87%
↑ vs. post-merge scanning baseline

COST PER CAUGHT VULNERABILITY
-73%
vs. production incident resolution cost

03

GASSIST

AGENTIC DEVELOPER ASSISTANT

In Slack. In Teams. Already there.

GAssist doesn't live in a tab developers forget to open. It operates as an agent in Slack and Microsoft Teams, answering technical questions, surfacing requirements clarity, resolving delivery blockers, and sending rich Adaptive Cards for build results and PR notifications. It draws on GKS to answer with organizational context, not generic AI responses.

When a developer asks about the auth flow architecture, GAssist traverses the knowledge graph: the ADR, the indexed Slack thread that shaped it, the modules that implement it. Answer in seconds. Source links included. Staleness flags where relevant.

SLACK NATIVE | MS TEAMS NATIVE | GKS-BACKED ANSWERS | ADAPTIVE CARDS (TEAMS) | PR NOTIFICATIONS | BUILD RESULT ALERTS | DAILY DIGESTS | BLOCKER RESOLUTION

BLOCKER-TO-RESOLUTION TIME
-54%
↓ vs. async email / ticket escalation

ANSWER ACCURACY VS. GENERIC AI
89%
↑ With GKS org-context grounding

04

G-IDE

IDE EXTENSION

Org-context, inside the editor you already chose.

For developers whose entire work life is their editor, context-switching to a browser or chat tool for an architecture question kills flow. G-IDE brings Garth's organizational context directly into the editor: code suggestions grounded in GKS and your team's established patterns, inline architecture answers without leaving the file.

Available in **VS Code, Cursor, Windsurf, and Antigravity**. Notably: Cursor and Windsurf are themselves AI-native editors. G-IDE layering GKS org-context on top of their general AI means something specific: your team's codebase knowledge, conventions, and architecture decisions amplifying the editor's built-in intelligence. A meaningfully different class of suggestion.

VS CODE | CURSOR | WINDSURF | ANTIGRAVITY | GKS-GROUNDED SUGGESTIONS | INLINE REVIEW FEEDBACK | ARCHITECTURE Q&A | TEAM PATTERN AWARE

CONTEXT SWITCHES ELIMINATED / DEV / DAY
8.4
↓ Interruptions to external tools

SUGGESTION ACCEPTANCE RATE
71%
↑ vs. 34% for generic completion models
Team-pattern awareness means suggestions fit, not just syntactically, but architecturally

---

05

G360

ENGINEERING INTELLIGENCE PLATFORM

Who are your AI leaders? Where is the spend going?

Engineering leaders are asked to justify AI tooling budgets without the data to do it honestly. G360 closes that gap, not just for Garth products, but for the entire engineering tool stack. Usage, adoption rates, cost attribution, and optimization signals, all in one view.

The metric that changes the conversation: **lines of code changed vs. tokens consumed.** For the first time, leaders can see which engineers and teams are shipping meaningfully with AI, and where token spend is high but code quality, acceptance rates, or output velocity suggests diminishing returns. Identify your AI leaders. Coach the rest. Optimize the spend.

And it reaches past AI entirely: DORA delivery metrics, CI/CD pipeline health, and cloud spend across AWS, GCP, and Azure. The same platform that tells you who is shipping with AI tells you whether delivery is getting faster, and what it costs to run.

NEW INTELLIGENCE LAYER

AI Leader Identification

G360 plots every engineer and team on a lines-changed vs. tokens-used matrix. High output, good token efficiency, strong code quality scores: those are your AI leaders. High token spend with low acceptance rates or recurring defects: that's a coaching signal, not a performance review.

LINES VS. TOKENS METRIC | AI LEADER IDENTIFICATION | TOOL ADOPTION TRACKING | COST ATTRIBUTION | SHELFWARE DETECTION | OPTIMIZATION SIGNALS | CROSS-TOOL COVERAGE | DORA DELIVERY METRICS | CI/CD PIPELINE ANALYTICS | CLOUD COST & ROI (AWS · GCP · AZURE) | EXECUTIVE DASHBOARDS | RENEWAL INTELLIGENCE

SHELFWARE IDENTIFIED (AVG. FIRST 30D)
28%
Of licensed tools with <10% adoption

RECOVERABLE / YR · 200-DEV ORG
$180K–$500K+
Floor to full-suite ceiling
$180k from shelfware rationalization alone; more as you layer in review efficiency, incidents avoided, and cloud FinOps.

06

GRELEASE

RELEASE INTELLIGENCE · IN PREVIEW

Ship on evidence. Not on the standup.

Every release tracker answers "are we on track?" with a status color someone set in a meeting. GRelease answers it with evidence. It joins **Jira commitment, Git landing, CI verification, coverage, and cross-team dependency state** into one graded readiness verdict per release, and it tracks every gate date, code freeze, verification, and release day, recording each slip as it happens. Grounded on GKS.

GRelease is in preview and launching soon. Explore the live preview ↗

DIFFERENTIATOR

Ask vs Built, reconciled

GRelease reads what each ticket asked for and what the code in its PRs actually built, then reconciles the two: delivered as asked, gaps still open, and the over-delivery hiding inside ticketed work that no burndown chart shows.

AFTER THE SHIP

Field quality feedback

Shipped releases keep reporting back. Customer-found issues are tied to the release that shipped them, grading whether it landed clean or rough, so readiness scores stay calibrated by what the field actually experienced.

COMMITTED VS LANDED VS VERIFIED | GRADED READINESS VERDICT | ASK VS BUILT RECONCILIATION | GATE DATES WITH SLIP HISTORY | CROSS-TEAM DEPENDENCY STATE | CI + COVERAGE SIGNAL | FIELD QUALITY ON SHIPPED RELEASES | JIRA · GIT · CI | GKS-GROUNDED

△ CONTROL PLANE
GOVERNS EVERY PRODUCT ABOVE

Garth Universe

Where a set of tools becomes a platform.

Universe is the portal you do not think about until you need all of it governed at once. It centralizes account, configuration, governance, usage, and audit across the suite. Set a policy once and it applies everywhere. Onboard an enterprise without standing up seven products by hand.

TENANTS
Companies, teams, environments, deployment mode.

USERS & ROLES
Admins, developers, reviewers, SSO groups.

LICENSES
Plans, seats, repo and PR limits, overages.

INTEGRATIONS
GitHub, GitLab, Bitbucket, Jira, Linear, Slack, Teams.

POLICIES
Review packs, protected paths, severity rules, AI guardrails.

AUDIT
Agent actions, config changes, approvals, usage logs.

WHY THE CIO CARES

One source of truth for entitlements and governance: least-privilege access, on-prem options, and a full audit trail of every agent action and config change.

PERSPECTIVE · CTO

The CTO

Is the architecture honest? Does GReview know what it doesn't know?

Stance after review

8/10

Convinced on architecture

My instinct with any AI review tool is the same as with a new hire: *what does it actually know?* A diff-reader that sees 40 lines without knowing those lines touch a critical auth pathway adds confident noise. That's worse than no review.

What shifts the calculus

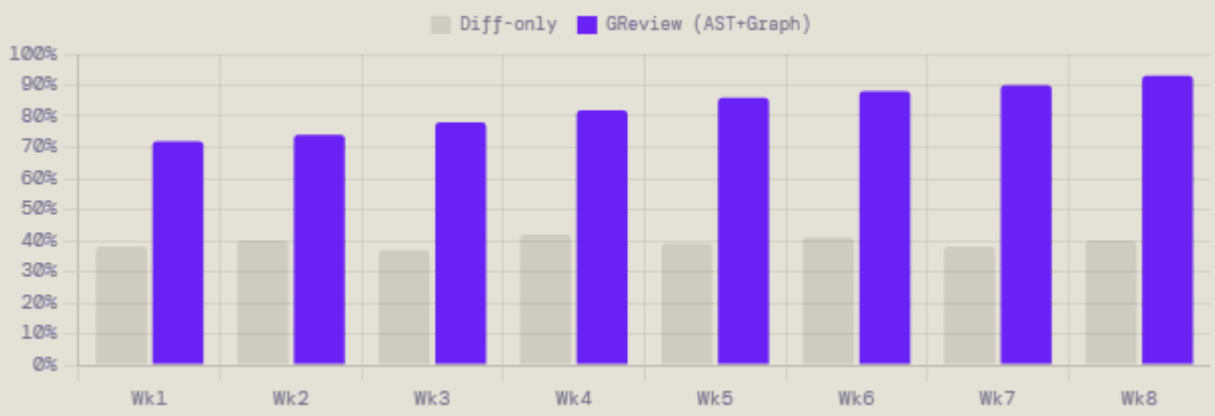
The inter-PR collision detection is the piece I didn't expect. We've had three significant integration conflicts in the last quarter, all of them invisible until merge day. If GReview surfaces those while both PRs are still in review, I get the conversation I needed three days earlier. That's not a nice-to-have.

The hidden agentic risk detection matters more every month. As AI-generated code enters our codebase at scale, I need to know when a contribution carries behavior I didn't authorize: unexpected side effects, non-deterministic patterns, prompt injection surfaces. Prompt Valuation on the roadmap closes the loop I've been trying to close manually.

> *"I stopped caring about AI tools that read diffs. I care about tools that read my codebase, watch my open PRs, and know my team's rules."*
>
> CTO perspective

GREVIEW COVERAGE BY CONTEXT DEPTH

SIMULATED, 60-DAY WINDOW



INTER-PR COLLISIONS SURFACED PRE-MERGE

94%

↑ vs. 0% with single-PR review tools

ARCHITECTURE VIOLATIONS CAUGHT

91%

↑ Cross-module + guardrail-aware detection

PERSPECTIVE · CFO

The CFO

What's the actual ROI, and how do I know which AI tools are earning it?

Stance after review

8/10

Numbers close in Q1

Every AI tool in our stack claims to save "hours per week." Nobody can show me the invoice. What I need is a cost model and visibility into what's actually being used, and G360's lines-changed vs. tokens-used metric finally gives me both in the same place.

Two things that close the math

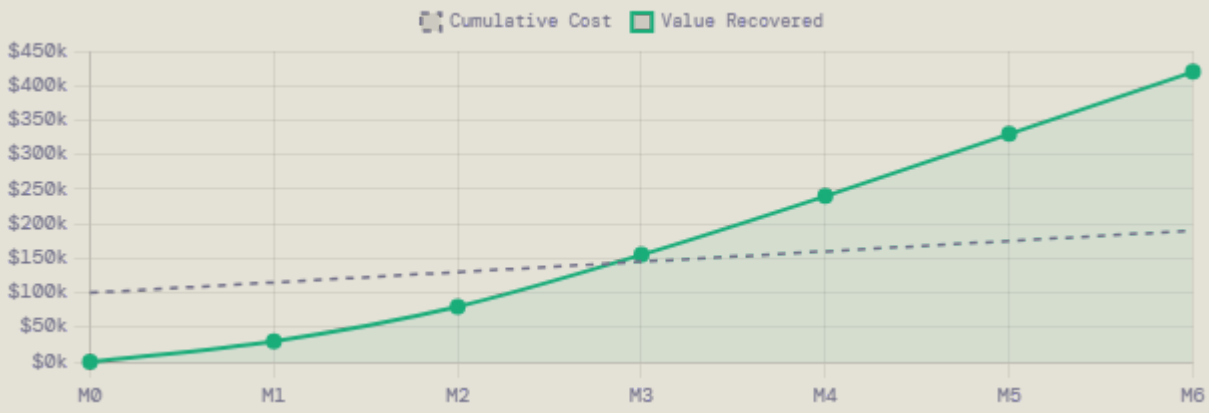
First, G360's AI leader identification. For the first time I can see which teams are shipping meaningfully with AI (high output, good efficiency) versus where token spend is high and the code quality data suggests we're burning budget without return. That's not a performance conversation. That's an optimization one. And I can have it with real numbers.

Second, shelfware detection. In 30 days, G360 surfaces which licensed tools have sub-10% adoption. Across a 200-person engineering org, that's typically $180k recoverable before the next renewal cycle. The G360 license pays for the entire Garth suite in recaptured budget alone.

And shelfware is the floor, not the ceiling. Stack it with GReview's review-time and defect-cost savings, GScan's incidents caught before production, and G360's cloud FinOps across AWS, GCP, and Azure, and total recoverable value runs $180K to $500K+ for a 200-person org in year one. It scales with the team and with how much of the suite you switch on: the more levers you enable, the harder the cost model closes.

COST RECOVERY MODEL · PER 100 DEVELOPERS

PROJECTED 6-MONTH WINDOW



PAYBACK PERIOD

1–3.2 mo

↓ From adoption to net positive

Unused team, server, and LLM licenses recovered in the first month; cloud FinOps and review savings compound over the quarter.

RECOVERABLE / YR · 200-DEV ORG

$180K–$500K+

Floor to full-suite ceiling

$180k from shelfware ID alone; the rest from review efficiency, incidents avoided, and cloud FinOps, by modules enabled.

PERSPECTIVE · CIO

The CIO

Where does the data go? Who controls it? What's the audit trail?

Stance after review

7/10

Satisfied on security

AI tools that introduce data exposure risks while "helping" are not new. My framework hasn't changed: where does data go, who controls it, what does the audit trail look like, and how do we manage access at org scale.

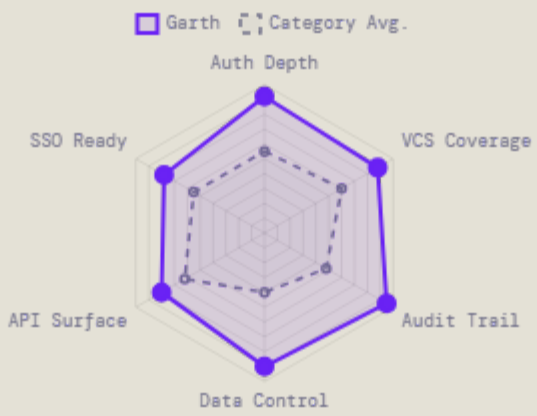
What passes the test

Ephemeral clone model for GReview: code doesn't persist beyond the review cycle. Dual-auth FastAPI (JWT + API key). OAuth user tokens on PR approvals so the audit trail carries real engineer identity, not a service account. GKS scoped by team and repo. It doesn't index what it isn't pointed at.

Garth Universe as the single control plane is the governance win I don't have to fight for. Configuration, licensing, access, integrations: one place, not six admin panels. When the security team asks where something is configured, there's one answer.

INTEGRATION SURFACE VS. ENTERPRISE CHECKLIST

GARTH VS. CATEGORY AVERAGE



AUTH MECHANISMS

4

JWT · API key · OAuth · GitHub App

DATA RESIDENCY

Full

GCP region configurable · ephemeral clones

PERSPECTIVE · ENGINEERING LEAD

Engineering Lead

Will my team use it? Does it reduce decisions or multiply them?

Stance after review

9/10

Advocate

My litmus test is ruthless: does this reduce the number of decisions my engineers make, or add new ones? I've lived through three AI tools that added process without removing friction. They're all gone.

Where Garth wins

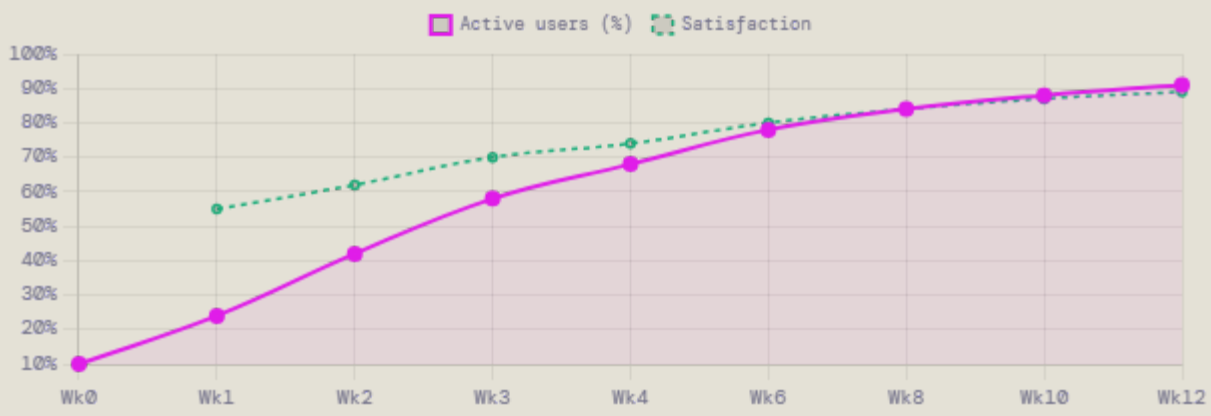
GReview done before I open GitHub. GAssist in the Teams channel answering the architecture question the new hire was too nervous to ask in standup. G-IDE suggestions that match how *we* write code in Cursor, not how the average GitHub repo does. The inter-PR collision alert that saved us from a merge day incident last month. These are friction removals, not additions.

The per-repo `.reviewconfig.yml` matters more than people realize. My payments team needs different guardrails than my tooling team. Garth doesn't impose a one-size model. It enforces *our* standards, per team, per repo.

> *"The best tool is the one that's already done its job before you knew you needed it."*
>
> Engineering Lead perspective

TEAM ADOPTION · WEEKS POST-DEPLOY

COMPOSITE, N=8 TEAMS



REVIEW-TO-MERGE CYCLE TIME

1.8h

↓ from 4.9h pre-Garth baseline

PIPELINE FRICTION SCORE

-38%

↓ Perceived workflow interruptions

The conversation every *leadership team* needs to have.

Real objections. Stitched by the PM. Resolved by the UX.

CTO — My concern isn't whether GReview works. It's what it does when it's wrong, and it will be. Does it fail loudly or quietly? And does the inter-PR detection create alert fatigue, or is it surgical enough to trust?

ENG LEAD — Loudly, by design. Risk tiers are explicit: High, Medium, Low, with the *reason* for every flag, not just the flag. The inter-PR alerts are scoped to actual shared symbol overlap, not proximity. We've had zero false-positive collision alerts in eight weeks. Engineers trust it because it earns that trust.

CFO — The lines-changed vs. tokens metric changes the renewal conversation entirely. I no longer have to take "our developers love it" on faith. I can see which teams are shipping efficiently with AI and where we're burning tokens without return. That's the data I need to defend, or cut, any tool in the stack.

CIO — Agentic risk detection is the piece I want to pressure-test. As AI-generated code enters our codebase at scale, *what does "hidden agentic risk" mean in practice?* What patterns trigger it, and how are they surfaced?

ENG LEAD — Non-deterministic output patterns, unexpected external call surfaces, prompt injection vectors baked into generated code, side effects that aren't expressed in the function signature. GReview flags these as a distinct risk tier (Agentic Risk) separate from standard code quality. Prompt Valuation on the roadmap closes the upstream loop.

CTO — G-IDE in Cursor and Windsurf is the move I didn't expect. Those editors already have strong AI. Layering GKS org-context on top means the suggestions are grounded in *how we build*, not how the average open-source repo does. That's a different category of assistance.

CFO — G360 surfaced $180k in shelfware in the first month. *That's the Garth suite paid for before the first renewal conversation.* The ROI story here isn't just about what Garth does. It's about what G360 tells us to stop paying for.

PM — Here's what I'm hearing: the CTO is convinced by the architecture and the inter-PR story. The CFO has a cost model that closes twice: GReview ROI and G360 recapture. The CIO has the security posture they need with one governance question still open. The Engineering Lead is already shipping with it. The missing piece is *what it actually feels like*: for the developer in the work, and for the leader watching it.

UX — *Two stories. Same day. Let me show you both.*

Where Garth stands.

Compared against tools that appear in the same shortlist conversations.

CAPABILITY	GARTH SUITE	CODERABBIT	GREPTILE	LINEARB	HIVEL
AST + GKS-grounded code review	✓ GReview	◐	◐	✗	✗
Inter-PR collision detection	✓ GReview	✗	✗	✗	✗
Team guardrail enforcement	✓ GReview	◐	◐	✗	✗
Hidden agentic risk detection	✓ GReview	✗	✗	✗	✗
Automated test generation for PRs	◐ Roadmap	✗	✓ TREX	✗	✗
Prompt Valuation	◐ Roadmap	✗	✗	✗	✗
Multi-VCS (GH + GL + BB + ADO)	✓ All 4	✓ GH+GL	◐ GH+GL	✓ All 4	✓
Repo security scanning	✓ GScan	✗	✗	✗	✗
Developer agent (Slack + Teams)	✓ GAssist	◐	✗	◐ WorkerB	◐ Slack
IDE: VS Code + AI-native editors	✓ G-IDE	✗	◐	✗	✗
Lines-changed vs. tokens-used analytics	✓ G360	✗	✗	◐ Copilot ROI	◐ Adoption
DORA delivery metrics	✓ G360	✗	✗	✓	✓
CI/CD pipeline analytics	✓ G360	✗	✗	✓	◐
Cloud cost & ROI (AWS · GCP · Azure)	✓ G360	✗	✗	◐	✗
AI leader identification	✓ G360	✗	✗	✗	✗
Cross-tool shelfware detection	✓ G360	✗	✗	✗	✗
Graph-based knowledge retrieval	✓ GKS	✗	◐	✗	✗
Central control plane + licensing	✓ Universe	✗	✗	◐	◐

> *Garth doesn't write your code. Copilot and Cursor do that. Garth governs, secures, and measures everything that happens after the prompt: the review, the risk, the delivery, the spend.*

ANALYST NOTE
MAY 2026

In May 2026, Gartner published its first *Magic Quadrant for Developer Productivity Insight Platforms*: formal recognition that this is now a defined enterprise buying category, evaluated against explicit criteria spanning delivery intelligence, AI adoption and ROI, engineering governance, and executive reporting. The capability surface Garth was built for is the surface analysts are now measuring.

Start anywhere. The value compounds.

You do not have to deploy the whole suite to get value from it.

Garth is a suite, not a bundle you have to swallow whole. Each product solves a complete problem on its own, so you start with the pain that hurts most today and expand on your terms. Nothing is wasted along the way: every product you add shares the same backbone, so the value compounds instead of fragmenting.

START WITH REVIEW

Govern every PR. AST + GKS aware, with inter-PR context so nothing slips between parallel branches.

STANDALONE

START WITH SECURITY

Catch security and quality issues before they ever reach main.

STANDALONE

START WITH VISIBILITY

Tie AI spend to delivery. Per-cloud FinOps, DORA, and adoption, each module enabled independently.

STANDALONE · MODULAR

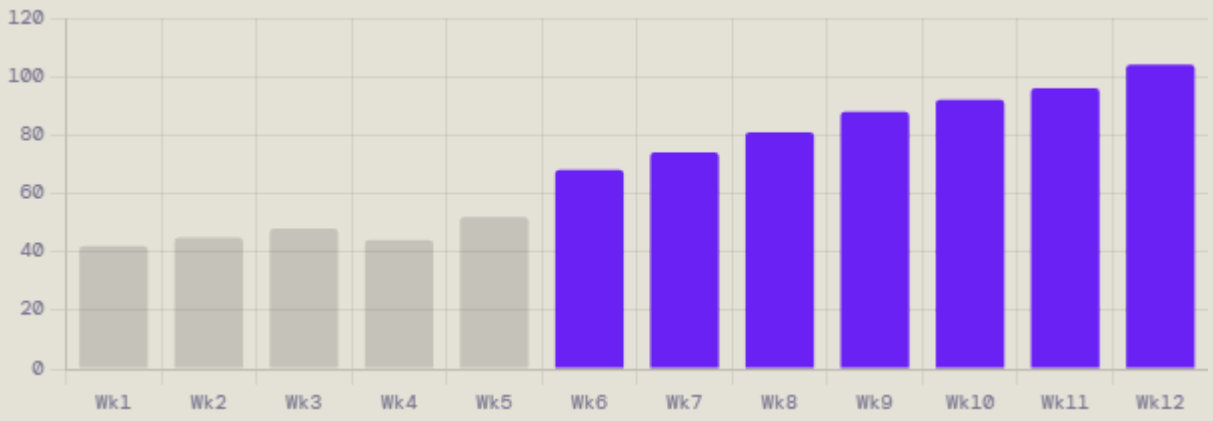
Then layer in what fits. GAssist answers in Slack, Teams, and your editor; G-IDE rides alongside GReview in any VS Code based editor. And GKS, the shared knowledge backbone, comes with every deployment by default. Each product you add gets smarter the moment it joins.

What the data suggests.

Simulated projections from early-access deployment patterns. Signals, not guarantees.

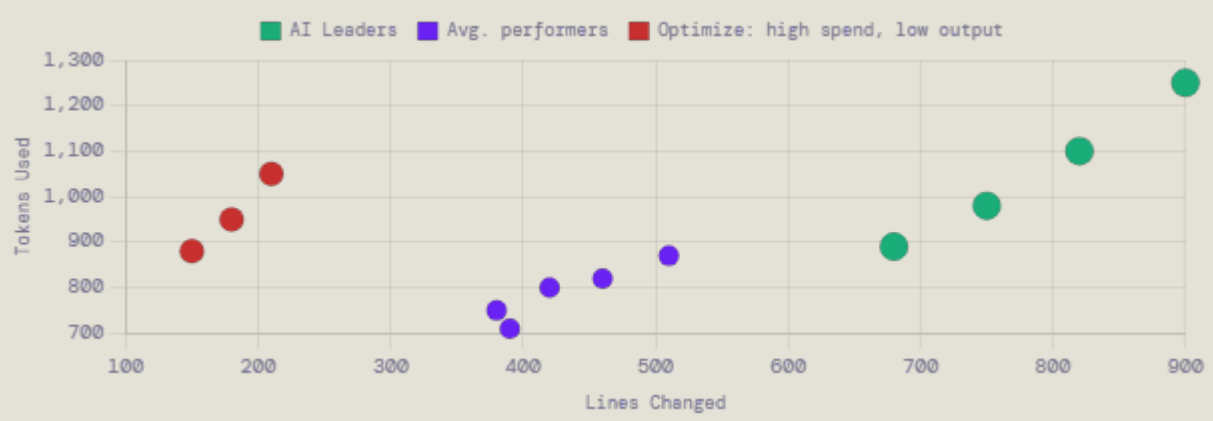
GREVIEW THROUGHPUT · WEEKLY

12-WEEK WINDOW, PRE VS. POST DEPLOY



G360: LINES CHANGED VS. TOKENS USED

TEAM EFFICIENCY SCATTER, 8-WEEK SAMPLE



PREDICTION · 6MO

67%

of repetitive review comments eliminated as GReview learns team-specific guardrail patterns

Horizon: Month 6, full GReview deployment

PREDICTION · 3MO

2.3×

knowledge retrieval accuracy as GKS graph density increases with team contributions

Horizon: Month 3, GKS + 2 connected teams

PREDICTION · 12MO

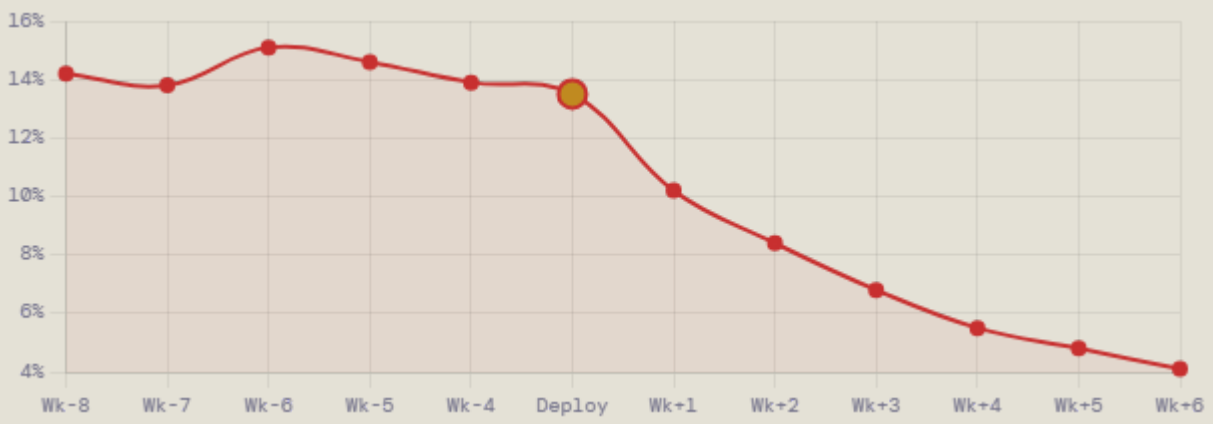
-61%

onboarding cost reduction as institutional memory becomes queryable via G-IDE + GAssist

Horizon: Year 1, consistent documentation input

DEFECT ESCAPE RATE · PRE VS. POST GREVIEW

ROLLING 4-WEEK AVERAGE



UX PERSPECTIVE

A day with Garth.

Two people. Same platform. Completely different altitude. The best measure of a tool is not what it does. It's how little you have to think about it doing it.

9am

Morning starts with context, not catch-up

GAssist sends a daily digest to Teams before standup: open PRs, review statuses, flagged builds from overnight, any inter-PR collision alerts. You walk into the meeting already knowing what broke and who's waiting. Standup is 8 minutes, not 22.

GASSIST

10am

A PR lands. GReview is already reading it.

An engineer opens a PR on the payments module. Before a human reviewer touches it, GReview has cloned the repo, built the symbol graph, run AST analysis, checked team guardrails, and scanned for active inter-PR collisions. Result: High-risk flag on a method called by 14 other modules. A guardrail violation on test coverage threshold. And, critically, a collision alert: another open PR is modifying the same auth utility on a parallel branch. The human reviewer spends 12 minutes where they'd have spent 45, and the merge-day incident doesn't happen.

GREVIEW

10:30

GScan surfaces a dependency risk before anyone merges

While the PR is in review, GScan flags a new vulnerability in a third-party dependency introduced on the same branch. Context-enriched: what it affects, CVSS score 8.1, the fix. The engineer patches it before the review is finished. Zero back-and-forth. Zero post-merge hotfix.

GSCAN

11am

The new engineer asks an architecture question

She's trying to understand why the auth flow works the way it does. She asks GAssist in Teams. It queries GKS and traverses the graph: the ADR from eight months ago, the indexed Slack thread that shaped the decision, the three modules that implement it. Answer in 11 seconds with source links and a staleness flag on one doc. She doesn't interrupt anyone. The answer is better than Confluence search on its best day.

GASSIST + GKS

2pm

A senior engineer builds. G-IDE builds with her.

She's writing a new endpoint in Cursor. G-IDE surfaces completions grounded in the team's established patterns via GKS, not generic training data. She accepts **71%** *of suggestions. She asks an architecture question inline. It answers without a browser tab. She stays in flow for four hours straight. No context switches. No interruptions.*

G-IDE IN CURSOR

4pm

The build breaks in staging

GAssist catches it instantly. An Adaptive Card lands in the team channel: the failing test name, the last commit that touched it, the PR that introduced the change, a link to the full build log. The engineer fixes it in **18min** *. Average without Garth context: 47 minutes. Nobody filed a ticket. Nobody pinged the senior engineer on Slack.*

GASSIST

5pm

End of day. Garth keeps working.

Three more PRs queued overnight. GReview is reading them. GKS flags two documentation entries as stale after the day's architectural changes. GAssist schedules tomorrow's digest. No engineer is waiting on anything tomorrow that Garth can prepare tonight.

ALL PRODUCTS

8:30

Before leadership standup: the picture is already clear

G360 sends its morning brief: AI adoption by team, yesterday's token spend vs. lines shipped, top movers, flagged anomalies. One team consumed 40% of the AI token budget yesterday. Their PR acceptance rate for AI suggestions: 18%. The benchmark is 65%. That's a coaching conversation, and it's identified before the CTO's first meeting of the day.

G360

9:30

A High-risk GReview escalation lands

G360 surfaces a GReview escalation: a High-risk PR on the payments module with an inter-PR collision alert and a guardrail violation on test coverage. The CTO sees it in the dashboard before the engineering lead has opened Slack. She sends a single message to the team lead. The situation is handled before it becomes a standup agenda item.

G360 + GREVIEW

11am

The board asks about AI tooling ROI. Answer: 3 minutes.

A board member asks for the ROI story on AI tooling investment ahead of next quarter's review. The CTO pulls the lines-changed vs. tokens-used report from G360: team-by-team efficiency, AI leader identification, GReview defect reduction data, and the $180k in shelfware identified in the first month. Three minutes. Board-ready. No BI team involved.

G360

1pm

New team onboarded. No IT ticket.

A new engineering team is onboarded onto the platform. Garth Universe provisions access, configures GReview guardrails for their repos, activates GAssist in their Slack channel, and sets G-IDE to their team's language and pattern preferences. The engineering lead does it herself in Garth Universe. No IT ticket. No back-and-forth. The team is productive by 2pm.

GARTH UNIVERSE

3pm

G360 surfaces an optimization signal

G360 flags an alert: one team is consuming **40%** *of the monthly AI token budget, but their GReview suggestion acceptance rate is 18% vs. a 65% benchmark, and their post-merge defect rate hasn't moved. The CTO schedules a 30-minute session with the team lead. Not a cost conversation. An effectiveness one. The data makes the difference.*

G360

4:30

CFO asks about renewals. G360 has the answer.

Renewal season is approaching. The CFO asks which tools to keep. G360 produces the cross-tool adoption report: two licensed tools with sub-10% adoption and renewal dates next month. Both cancelled. **$180k** *recaptured. The CFO signs off on the Garth renewal without a line of negotiation. The ROI is in the report.*

G360

5pm

End of day. The platform runs without you.

GReview is processing overnight PRs. GKS is flagging stale documentation. GAssist is preparing tomorrow's team digests. G360 is computing the next morning's efficiency brief. The CTO didn't manage any of it. The platform did. That's what infrastructure feels like.

ALL PRODUCTS

What Garth carries for your team is not *a feature.* It's a habit.

THE CTO'S TAKE

Architecture-honest. GReview reads the codebase, watches open PRs, enforces guardrails, and flags agentic risks. That's the right unit of analysis. G360 tells me what my teams are actually getting from AI, in numbers I can defend.

THE CFO'S TAKE

The cost model closes twice. GReview ROI from defect reduction. G360 ROI from shelfware recapture. Modular licensing means I don't bet the budget before I see results. The lines-vs-tokens metric is the most honest AI ROI measure I've seen.

THE TEAM'S TAKE

The first suite that made standups shorter, flow states longer, merge-day incidents rarer, and the build break at 4pm less painful. Those are the measures that matter to the people doing the work.