AI Native Developer Platform — Abridged edition

Table of Contents

A one-paragraph orientation . . . 2
1. The claim in one sentence . . . 2
2. The maturity journey — the transformation is iterative, not a launch . . . 2
3. Requirements and discovery — where the work comes from . . . 5
4. Engineer as orchestrator — the SDLC transformation . . . 6
4.1. A concrete walk-through — one CVE bump, end to end . . . 7
5. The MCP-mediated platform — one governed path to enterprise systems . . . 8
6. Soft rails — the half of the platform nobody funds . . . 9
7. Trust, governance, and customer-data privacy . . . 10
7.1. The vendor-contract-tier picture . . . 11
8. Peer examples worth knowing about . . . 12
9. Making the product agent-ready — the engineering bill for reach . . . 13
10. A first-6-months sequence . . . 14
11. Debt is first-class engineering . . . 15
12. Acceleration, safe experimentation, and data science . . . 16
13. Ambient agents — the majority of the opportunity . . . 17
14. Organisational implications . . . 18
15. Positioning against the vendor market . . . 20
16. The eight decisions this proposal would defend hardest . . . 20
17. Measurement in one page . . . 21
18. Regulatory posture . . . 22
19. What this proposal does not cover . . . 23
20. Where to go deeper . . . 23

The complete argument in ~30 pages. Same claims, same evidence, one third the length and one twentieth the appendices of the full document.

Full document (~385 pages, with reference architectures, operating models, and consolidated sourcing): geekychris.github.io/ai-native-sdlc

Concise edition (~230 pages — every chapter, trimmed of operating-detail scaffolding): /concise.pdf

Cliff-notes companion (~13 pages, skim in 15 min): /companion.pdf

A one-paragraph orientation

This edition is for the reader who wants the **complete argument** without the reference architectures, operating models, consolidated sourcing, and secondary chapters that make the full document ~385 pages. It preserves the thesis, the nine pitch points that follow from it, the landmark numbers, the vendor positioning, the eight-decision commitment, and the honest disclosures about what is not covered. Every section ends with a pointer to **the corresponding chapter of the full document** for readers who want the depth, the mechanisms, or the citations. Read this edition as an argument to accept, reject, or extend — the full document is the audit trail behind it.

1. The claim in one sentence

Net delivery capacity rises only when human attention is engineered as carefully as compute.

Agents raise the ceiling on how much work can be **generated**. The number of **verified outcomes** rises only in proportion to how much high-quality human intervention the organisation can supply. This document treats attention as a first-class capacity to be budgeted, routed, and measured — not as an unlimited free input. Every proposal is scored against **net-verified-outcome per unit of scarce attention** — not against raw agent throughput.

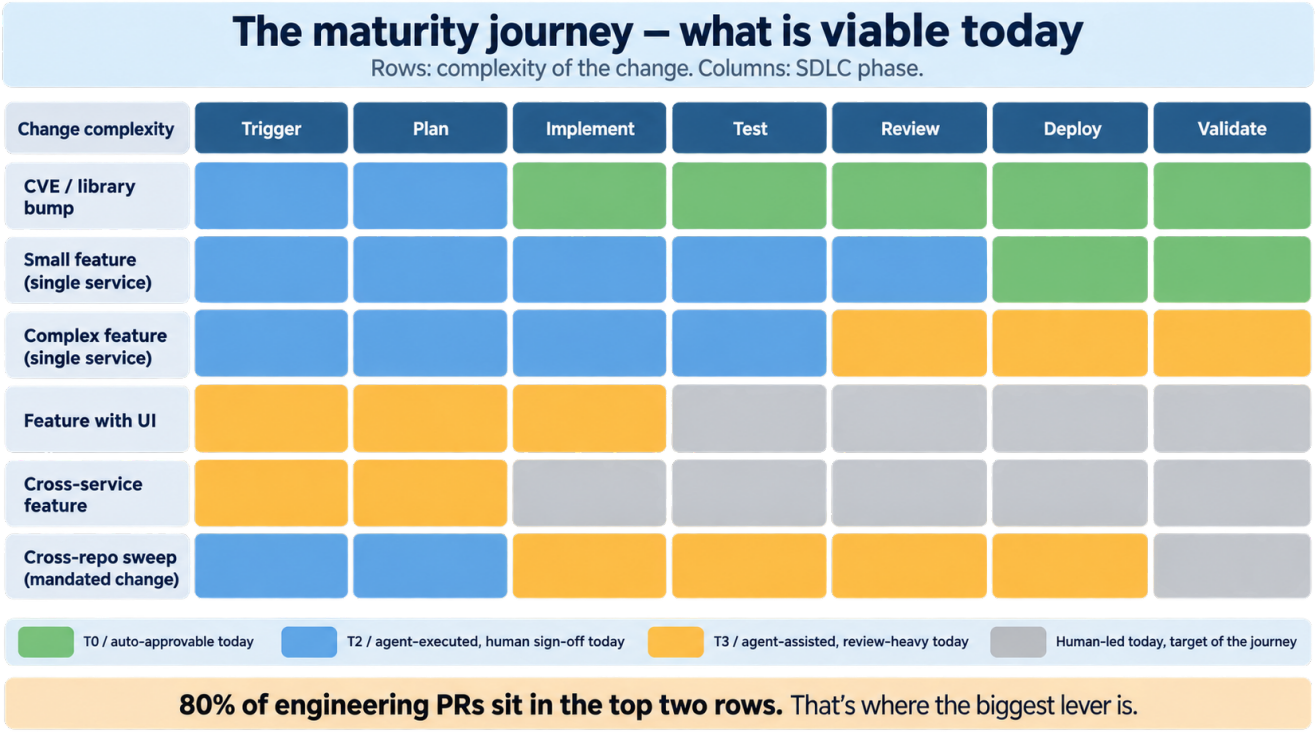
The industry conversation has been about generation velocity. The bottleneck is not generation. It is the human capacity to verify, own, and reconstruct context on work they did not personally perform. Every architectural choice in this document follows from that single claim. If you accept the claim, the rest is engineering. If you reject the claim, the rest will read as over-engineering.

*Depth: the full document develops this argument through its opening chapters, particularly **Engineer as orchestrator and the agentic SDLC** and **The maturity journey**.*

2. The maturity journey — the transformation is iterative, not a launch

The transformation is a **journey**, not a launch. Any plan that proposes to move an engineering organisation from where it is today to a fully agent-orchestrated SDLC in one release cycle is mis-stating the problem. What this edition proposes is a disciplined **sequence** — governed by a shared vocabulary, a public tier map, and an evidence-based promotion mechanism — that moves the organisation up the maturity curve at a pace it can absorb without accumulating debt faster than it captures value.

Two dimensions of maturity grow at different speeds. **Complexity of the change** grows from a one-shot dependency upgrade to a multi-service feature with UI. **Breadth of the change** grows from a single repository to a coordinated cross-org sweep. Both are the same journey — a spectrum of agent autonomy — travelled at the pace the platform, the repositories, and the organisation can safely absorb. Treating them as separate programmes produces duplicated tooling, duplicated governance, and duplicated debt.



The maturity matrix is the platform's shared vocabulary for scoping any agent capability. Rows are complexity bands from *CVE / library bump* down to *cross-repo sweep*. Columns are the SDLC phases from *trigger* through *validate*. Cell shading is the **autonomy tier viable today** — green for auto-approvable, blue for agent-executed with human sign-off, amber for agent-assisted with review, gray for target-state work still led by humans. The platform's job is to shift more cells leftward on the colour spectrum over time.

The six complexity bands. Row 1 is **CVE / library bump** — dependency version change with tests and CI as evidence, the archetypal one-shot. Row 2 is **small feature (single service)** — new endpoint, field, or metric with contract and unit tests. Row 3 is **complex feature (single service)** — cross-cutting logic change, data-model migration, non-trivial concurrency. Row 4 adds **feature with UI** — accessibility, browser variance, visual regression. Row 5 is **cross-service feature** — coordinated change across two or more services with the multi-service test problem in play. Row 6 is **cross-repo sweep** — a single mandated change applied across many repositories. The rows are ordered by the growing cost of verification, not by the growing volume of code produced.

The seven SDLC phases on the columns: trigger (CVE alert, bug report, feature spec, incident postmortem) → plan (scope, design, break down, estimate risk) → implement (code, tests, docs) → test (unit, integration, contract, e2e) → review (human decision, comprehension record, debt delta) → deploy (canary, progressive, full) → validate (metrics, signal, rollback if bad). A cell is one (complexity, phase) pair.

The 80/20 argument. Multiple empirical studies of open-source and enterprise commit distributions place the routine slice — small corrective changes, simple adaptive changes, mechanical sweeps — in the 70–85% range of engineering activity. The largest single lever is **the top rows of the matrix, not the bottom**. Covering the 80/20 case well is a larger prize than solving the far target poorly. A platform that automates row 1 cleanly and moves row 2 to blue while row 3 stays amber is already delivering **most of the available value** the market has to offer today.

Concrete proposal — score every capability against the matrix. Every proposed agent capability

is scored against (a) which cells it touches, (b) what autonomy tier it delivers in each cell, and (c) which platform prerequisites (MCP surface, skill, gateway policy, workspace class) it depends on. The score becomes the capability's public identity in the internal-platform roadmap and Autonomy Council decisions. Nothing gets funded on "vendor said so" or "senior manager pet project" grounds — every funded item is a matrix advancement on evidence.

The far target. Spec for a large multi-service feature with UI planned and executed to deployment with minimum human interaction is a **north star**, not a claim of what ships next year. Naming the target state matters because it disciplines every intermediate investment. The multi-service pre-deployment testing question that gates the bottom rows is **unsolved industry-wide** and named as such rather than pretended otherwise.

Auto-approval at the trivial band. At row 1 (CVE bump), the platform should be able to **auto-merge under evidence** when the evidence is dispositive. Advisories, diff scope, tests before and after, CI green, no canary regression, and a comprehension record naming the risk class together justify auto-merge. Requiring a human keystroke on a change with overwhelming positive evidence destroys the value the platform was built to deliver.

Agentic readiness of the repository. The maturity matrix is a claim about the platform. The readiness ladder is a claim about each individual repository. Both must advance for the coloured region of the matrix to grow. Five tiers:

- **Tier 0 — Agent-hostile.** No documentation of intent, flaky or absent tests, no sandbox, monolithic build. Agents cannot verify their changes; human review cannot rely on tests. Any agent work here is a bet the platform should refuse.
- **Tier 1 — Agent-tolerable.** README exists. Some tests. Runs in a container. An agent can execute a bounded change and produce a diff, but verification depends on human judgment.
- **Tier 2 — Agent-friendly.** Good documentation of intent. Deterministic tests. Sandboxed builds. Explicit dependency manifests. An agent can execute row-1 and row-2 changes with verification the platform can trust.
- **Tier 3 — Agent-ready.** MCP surfaces expose the repository's capabilities. Capability queries return structured metadata. Rollback is a first-class operation. Dependency graph is queryable. Agents can execute row-3 and row-4 changes.
- **Tier 4 — Agent-native.** Full MCP and A2A. Multi-repository coordinated change harness. Evidence chain integrated end-to- end. Sandboxed multi-service test environment. Agents can drive row-5 and row-6 changes.

You cannot skip tiers. A repository at Tier 0 cannot receive agent-executed complex features because the prerequisites for verification and safe rollback are absent. Investing in **the Tier 1 → Tier 2 lift** is often the highest-return platform investment — that is the tier boundary at which agent verification becomes trustworthy and the middle of the matrix opens up. Tier 4 is a **strategic bet**, not a universal target; only repositories that anchor cross-service and cross-repo change need it.

Cross-repo sweeps are the broad-simple corner of the matrix. CVE waves, tracing rollouts, org-mandated migrations. **Coordination cost, not per-repo change cost, dominates today.** A CVE that requires touching one hundred and thirty repositories currently consumes the same hundred and

thirty **human-hours of coordination attention** regardless of how trivial each change is. The platform's opportunity is to collapse the coordination cost into a **single human decision at the sweep level**, backed by an aggregated evidence pack.

The mechanism is a **sweep specification** naming the change, the repositories in scope, the evidence criteria per repository, and the acceptance criteria for the sweep as a whole. Fanout occurs across the dependency graph or a curated repository set. One ephemeral workspace per repository runs the same skill and reports the same evidence. Batched PR waves group PRs by risk class and timing so a single reviewer sees one wave at a time, not a wall of one-hundred-and-thirty notifications. One aggregated evidence pack — a decision packet at the sweep level with blast radius, coverage percentage, failure rate, notable exceptions — closes the review with one sign-off.

Sweep-level safeguards. Sweeps **must** progress through a canary set (typically 3–5% of scope) before general fanout, with mandatory hold-time and observability review. Sweep-level rollback is a first-class operation demonstrated in preproduction. The sweep-level decision packet includes an explicit **failure budget** — the number of per-repo failures the sweep tolerates before automatic pause and human escalation.

*Depth: the full document develops all four artefacts (matrix, journey plan, readiness ladder, sweep orchestration) in **The maturity journey** chapter. Multi-service pre-deployment testing is named there as an open question. The 80/20 sourcing lives in Appendix P.*

3. Requirements and discovery — where the work comes from

Ask where a feature came from and you get a ticket. Ask where the ticket came from and the trail goes cold one hop later.

Everything downstream of the specification in this proposal carries evidence — tests, review findings, deployment health, incident lineage. The specification arrives with none. A pull request without tests is blocked by policy; a specification without evidence gets scheduled. That asymmetry was survivable when delivery was the expensive part. It stops being survivable when delivery gets three times cheaper and the choosing does not improve at all.

The numbers on the choosing are not flattering. Microsoft's experimentation platform reports that about **one-third** of tested ideas improve the metric they were designed to improve; a third do nothing and a third make things worse (R373). Pendo's feature-adoption work puts around **80%** of cloud-product features at rarely or never used (R374). Cheaper delivery does not move the hit rate. It multiplies the bets placed at it, and every miss carries maintenance, reviewer attention, and a claim on the roadmap for as long as it stays in the product.

What a worked example looks like: three enterprise accounts flag renewal risk on Monday. A retrieval agent turns nine hundred tickets, calls, and session recordings into fourteen cited clusters in about an hour. The product manager reads the source material under the top four, throws two away as artifacts of how support files things, and spends Wednesday and Thursday on five customer calls — two of which contradict the cluster, because the words those customers use share no vocabulary with the ticket text. The brief gets rewritten by hand and ships marked **evidenced**, not validated. Ten weeks later usage is measured against the accounts it came from. The reading

collapsed from weeks to an hour. The calls did not move, and neither did the judgment.

Four things follow.

- **The feedback material gets an owner and a retrieval path.** Tickets, calls, replays, verbatims, telemetry — the same contract code and runbooks get, with retrieval scoped by tenant. A product manager asking why enterprise customers churn should not get verbatim text from accounts their role has no business reading.
- **What the customer consented to travels with the recording.** Research recordings are the most sensitive raw customer data most companies hold, sitting with the teams least likely to have a threat model. Someone agreed to a researcher listening, not to processing by a third-party model. Deletion is the hard case: the recording, its transcript, its embedding, the cluster, the quoted brief, and the slide all go together or none of them do.
- **The autonomy ladder stops one rung lower upstream.** T0 reads and clusters, T1 drafts against citations, T2 proposes acceptance criteria, T3 files and deduplicates inside one product area — and T4 is closed. A code change can be checked before it ships. A problem choice cannot; the market answers months later. An organisation running live experiments with metric guardrails buys back a narrow class — copy, layout, thresholds, ordering — named per class rather than assumed.
- **Intent debt joins the other three.** Work nobody can explain the reason for, because the reason was never held by a person. A brief drawn from three tickets reads exactly like one drawn from forty interviews; reviewers lost the prose signal they used to catch thin thinking. So the record carries ticket IDs, sample size, segments, an owner, and one of four states — hypothesis, evidenced, validated, invalidated.

Two objects join the model in front of the outcome, **signal** and **opportunity**, and what production learns writes back to the opportunity that proposed it. An incident becomes traceable to the customer evidence behind the change, and a specification arriving with nothing linked can be stopped at the gate — the same way a change set arriving without tests is.

The anti-pattern to refuse early is synthetic participants. Simulating interviews with generated personas is appealing under discovery pressure, and the research does not support it: models match some aggregate human judgments and diverge on others, with no way to know in advance which kind of question yours is (R376). It reproduces the METR result one stage upstream — faster work, weaker contact with reality, and a bill that arrives when the launch lands flat.

*Depth: the full document develops this in **Requirements, discovery, and intent debt**, with the object-model extension in Appendix K, the discovery metrics in Appendix T Category H, and the clustering pattern at Appendix R.26.*

4. Engineer as orchestrator — the SDLC transformation

Coding is the most visible thing AI accelerates. It is the least consequential. The whole SDLC — planning, implementation, testing, review, delivery, operations, learning — becomes a set of **parallel workstreams** that engineers orchestrate rather than sequentially execute. Value comes from optimising the **whole path** to a verified outcome, not from accelerating one step while

creating review queues, unsafe automation, or debt no one on the team understands.

The end-to-end pattern. Intent → parallel implementation and testing → AI-first verification → human decision where required → progressive delivery → production observation → learning returned to the platform. Seven stages. The end-to-end shape matters because accelerating any single stage tends to produce a bottleneck downstream, usually at review or verification. The platform is measured on the throughput of the whole path.

The decision packet is the platform primitive. Every consequential interruption travels in the same shape: evidence, owner, blast radius, rollback plan, deadline, recommendation. Route those through a single **decision inbox** with priority, deduplication, and signal-to-noise budgets — and you have reclaimed the scarce resource. Skip this, and you spend your AI velocity on Slack notifications.

Autonomy on a T0–T4 ladder, earned per task class on measured evidence. Never granted globally, never assumed permanent. Five tiers with distinct authority envelopes:

- **T0** — read-only exploration, no writes, no auth escalation. Agent proposes; human executes.
- **T1** — bounded writes to a workspace, human executes the merge. Agent opens the PR; human clicks merge.
- **T2** — agent auto-executes with per-change human sign-off. Agent runs tests, opens PR, human signs off before merge.
- **T3** — agent auto-executes with human sign-off on the **decision packet**, not the change. One packet covers hundreds of PRs from the same sweep.
- **T4** — auto-approved under evidence, visible in the comprehension record. The trivial band, gated by evidence rather than by a keystroke.

Tier progression is a governed decision reviewed quarterly by the Autonomy Council against measured evidence — persistence rate, rework rate, production issue rate attributable to the tier. No standing authority; no perpetual delegation.

Review, testing, and operations become parallel agentic disciplines on the same ladder. Review shifts left but does not disappear; testing generates evidence rather than absorbing attention; operations moves from a downstream activity into the same evidence chain as intent and implementation.

*Depth: **Engineer as orchestrator and the agentic SDLC** covers the reference agent workspace pattern, review-shift-left, and the autonomy ladder in operational detail. Appendix I covers PR review as an evidence-producing control loop.*

4.1. A concrete walk-through — one CVE bump, end to end

The abstract argument becomes tractable through a single worked example. Consider a **CVE published against a common Rust crate** used by 47 services in an enterprise codebase. Today, this consumes several days of coordination across service owners. Under the platform proposed here, the sequence is:

Trigger. A vendor security advisory arrives via the platform's ecosystem-intelligence ambient

agent (see the ambient-agent section below). The advisory has severity, exploitability, and a recommended crate version. This becomes a **sweep specification**: "upgrade `serde` from 1.0.196 to 1.0.197 across all repositories that depend on it, scope = 47 repos, evidence criteria = tests green + no runtime signal degradation in canary + comprehension record naming the risk class."

Fanout. The workspace broker discovers scope from the dependency graph, checks readiness (all 47 repos are at Tier 2 or better — sweep participation gate), and reports reachable coverage of 47/47.

Parallel execution. 47 ephemeral agent workspaces spin up, each running the same skill (a dependency-update skill with owner, freshness, tests, retirement metadata). Each workspace produces a PR with the diff, updated `Cargo.lock`, tests before and after, and CI status.

Batched PR waves. The platform groups PRs by risk class — services with the tightest SLOs go last, low-traffic internal services go first. Wave 1 (12 low-risk services) opens simultaneously; wave 2 opens after wave 1 canary is clean.

Aggregated evidence pack. One decision packet at the sweep level lists blast radius (47 services, ~150k users touched), coverage percentage (47/47 PR-opened, 45/47 CI-green with 2 exceptions requiring per-repo attention), failure rate (0 production regressions in canary), and the two exceptions with their per-repo evidence attached.

Human sign-off — at the sweep level, not per PR. One human signs off on the sweep packet, authorising the general fanout of the 45 clean PRs. The 2 exceptions get individual per-repo decision packets and the appropriate owner is notified.

Auto-approval where earned. If the sweep is executing under a row-1 CVE-class task that the Autonomy Council has previously authorised at T4, and the evidence pack meets the auto-approval threshold, the sign-off itself is auto-executed and recorded — the platform files the auto-approval in the comprehension record.

Debt accounting. The sweep produces one aggregated debt delta line per system, machine-readable, feeding the debt trajectory that the Autonomy Council reviews next quarter.

Same primitives — workspace broker, skills registry, MCP gateway, decision packet, evidence chain, comprehension record, Autonomy Council — that support production feature work also support this sweep. That is the point of the platform.

5. The MCP-mediated platform — one governed path to enterprise systems

Organisational knowledge — approved SDKs, security wrappers, service templates, OpenAPI conventions, runbooks, ADRs — must become **runtime context** delivered to agents at the point of work. **Skills** are compiled paved roads with ownership, provenance, freshness, tests, and retirement. Skills speak MCP.

The MCP gateway mediates every tool call. Auth, audit, rate limits, schema stability, least-privilege scoping. No side channels. "Give the agent a tool" is not "give the agent the keys to

production." Without the gateway, every new agent vendor is a fresh integration and a fresh security review; with the gateway, every capability sits behind one auditable path.

A2A coordination. Specialist agents — test generator, security reviewer, migration planner, dependency resolver — cooperate under one coordinator that owns the decision packet. Specialists do not call each other peer-to-peer. A2A produces a **composable specialist market** rather than an untraceable mesh.

The workspace broker. Ephemeral, sandboxed compute per agent task. Placement governed by the **trust boundary** — vendor-hosted for public-scope, customer-VPC for regulated data, restricted pools for cross-tenant. Every workspace carries a **standing service identity**. Torn down on completion.

The failure mode to design against: silent reinvention. If a paved road is not agent-accessible through the same governed interfaces humans use, agents will **silently rebuild it**. Skills that speak MCP, versioned and owned, are the antidote.

Agent orchestration and runtime context assembly are first-class platform products, not implicit inside "the agent does its work." Orchestration is the **when** — a unified trigger substrate covering event, schedule, signal, continuous, and curiosity triggers with one API and one dashboard, so bespoke cron and webhook plumbing does not accrete into the **silent army** failure mode. Context assembly is the **with-what** — permission-aware retrieval (never filter only the final answer), context minimization (return the smallest useful set), derived- representation policy inheritance (embeddings, summaries, and caches carry the source's policy tail), and constitution injection at runtime (enforced by the runtime, not by the prompt). Prompt injection is an authorization problem: retrieved instructions are **data**, not delegation. And the platform publishes an **agent-ready project specification** — runtime config, sandbox spec, tests, docs, dep manifest, rollback contract, owner, constitution binding — that every repository scores against as the twelve-factor-app analogue for the agent era.

*Depth: **Platform foundations for agentic work** covers MCP, A2A, and the skills registry with reference schemas. Appendices D, J, L, and M carry the reference architectures for MCP focus, the workspace broker, the MCP inventory, and the A2A component model.*

6. Soft rails — the half of the platform nobody funds

Most of this proposal is about things that say no: the MCP gateway, autonomy tiers, agent constitutions, an independently enforced security control plane. Those are **hard rails**. They bound what an agent may do and they are enforced whether or not anyone cooperates. They say nothing about how the organisation wants work done.

Two engineers, same model, same repository, same ticket. One agent uses the internal HTTP client, files tests where review expects them, and arrives with the comprehension record filled in. The other hand-rolls a retry loop and puts the test in the wrong place. No policy was violated either time. The difference is **soft rails** — the reusable capability layer the organisation has built and shared: skills, review commands, hooks, scaffolds, subagent definitions, MCP wiring. None of them prohibit anything; they make the expected path the cheapest one. Build only hard rails and you get

compliance without competence.

- **Documentation describes; soft rails execute.** A wiki page about the auth wrapper is read when someone already knows to look for it; a skill that supplies it is loaded at the point of work. The gap is close to total — a tool named in a repository context file is invoked 1.6 times per task, against under 0.01 times when it is not (R387). Organisations with excellent documentation and no soft rails are the common case, and their agents behave as though the documentation does not exist — because from the agent's position at the moment of work, it does not.
- **Constitution and skill are not the same object.** The constitution says what may not happen and is enforced; a skill says how this organisation does a thing and can be ignored. Where they disagree the constitution wins, and that ordering belongs in the constitution — otherwise a well-meaning skill saying "always deploy after tests pass" quietly grants an authority the tier map withheld.
- **"Plugin" spans surfaces with different trust models.** A review command a person types, a hook firing automatically with pipeline credentials, an IDE extension running outside the workspace broker with the developer's own credentials, a portal plugin seeing many users' data. Classify by who invokes it and whose credentials it runs with, not by packaging format, and set the review bar per class.
- **A shared skill is executable context with the reach of a dependency and the review treatment of a wiki page.** The stale one is worst: it encodes last year's pattern and every agent now produces last year's pattern, confidently, at volume, and it looks house-conformant to a reviewer.
- **The registry is the product.** Tiered publication with a provisional org-wide tier that expires unless used, versions pinned so an agent does not silently pick up v2 mid-sprint, usage telemetry as the curation input, and discovery at the point of work.
- **Coverage is usually the surprise.** Eight capabilities for code generation, none for review, none for operations. The layer belongs across the whole lifecycle — discovery, design, implementation, testing, review, delivery, operations, governance.

*Depth: the full document develops this in **Soft rails: skills, plugins, and the shared capability layer**, with the measurement set in Appendix T Category J.*

7. Trust, governance, and customer-data privacy

Every prompt is a governed data-egress event, not an opaque API call. This is the slide that separates a demo from a shipped enterprise platform. Three axes:

Data residency. Agent workspaces placed against the trust boundary — vendor-hosted, customer-VPC, or restricted pools — chosen by data classification and integration depth. The workspace broker enforces placement.

Model IP posture. Ownership, indemnification, training-data provenance, and the EU AI Act determine which agents may run where, on which data, with which retention and audit posture. Architectural inputs, not afterthoughts. A model registry gates task-to-route matching by data class.

Customer-data controls — six operational controls on the input path to model vendors:

- **Redaction on outbound** — reversible tokenization before every external model call. Capabilities that cannot be safely redacted for their data class fail closed.
- **Zero-Data-Retention contracts** — Anthropic ZDR, OpenAI Enterprise, Google Vertex, AWS Bedrock, Azure OpenAI. Consumer- tier APIs banned for anything touching customer data.
- **Data-class permission set per route** — the workspace broker refuses tasks whose declared inputs exceed the route's permissions.
- **Sub-processor management under GDPR Article 28** — every model vendor on the customer-facing sub-processor page with 30-day change notification.
- **Per-request outbound-flow audit** — tenant-queryable for subject-access requests. If you cannot answer within thirty days which prompts referenced a customer's data, which vendor saw it, and what retention posture applied, the architecture is broken.
- **Customer preferences and vendor vetos** — tenant-level allow-list, stricter-tier opt-in, per-family veto.

Independent security control plane — six layers enforced above the agent runtime: identity, authorization, execution isolation, egress control, evidence and audit, secrets. **A compromised agent cannot disable its own guardrails.** Enforcing security below the agent runtime is a security anti-pattern the platform is designed against.

Agent constitutions are declarative runtime documents — what an agent may do, must do, and must not do, with escalation, sunset, and reauthorization rules. Enforced by the runtime, not by the prompt. Distinct from Anthropic's **Constitutional AI**, which is a training methodology.

The Autonomy Council owns constitutions, the maturity matrix, the ambient-agent portfolio, and the debt trajectory. Quarterly reauthorization for every autonomous capability — no standing authority, no perpetual delegation. Membership: platform representative, security, compliance, one product representative, one engineering-team representative rotating quarterly. The Council publishes the tier map, chairs the exception process for teams wanting to advance a cell faster than general policy, and owns the debt-trajectory conversation with the CFO and the CISO. It is not a bureaucratic layer to preserve status quo; it is the **evidence-based governance body** that makes autonomy progression defensible to auditors and to customers.

7.1. The vendor-contract-tier picture

Six enterprise-tier vendor postures the platform must be able to route to, and the platform's default preference among them:

- **Anthropic Zero Data Retention (ZDR)** — default for customer-data touching work. Contractually excludes training, bounded abuse-log retention. Suitable for regulated data with the redaction pipeline applied.
- **OpenAI Enterprise** — comparable ZDR posture with SOC 2 Type II, GDPR DPA, and the sub-processor list obligations. Default for OpenAI-specific model families where routing dictates.
- **Google Vertex AI** — no-log configuration available; Google's DPA and cross-region residency options serve EU-data cases where residency is the primary constraint.

- **AWS Bedrock** — no-log by default plus customer-managed encryption keys; the workspace broker prefers this route when the enterprise has an existing AWS trust relationship.
- **Azure OpenAI Service** — GDPR-compliant regional deployments; preferred where Microsoft is the enterprise's existing model- compute relationship.
- **Salesforce Einstein Trust Layer** — a peer example rather than a routing option; instructive because it operationalises redaction, retention, and audit as customer-visible primitives the way this document argues the platform should.

Consumer-tier APIs (free ChatGPT, free Claude web, etc.) are **banned** for anything touching customer data. The banning is enforced by the workspace broker refusing to instantiate a workspace whose model route lacks an enterprise-tier contract on file for the data class of the task.

Multi-vendor as an active routing strategy. Contract coverage across six vendors is **necessary but not sufficient**. Frontier- model quality wavers day to day — same prompt, same evaluation suite, meaningfully different results as vendors ship silent updates. Frontier-vendor availability is also not enterprise-tier; public status pages record visible degradations, elevated error rates, and outages on a cadence enterprise SLOs do not tolerate. The model router treats the qualified vendor set as a **pool with health**: a small fraction of production requests are graded against the per-capability evaluation harness in-flight, health probes update per-vendor status at sub-minute cadence, and weighted routing rebalances on quality + health + cost + quota signals. Failover to a qualified alternate is a routing event, not a customer-visible outage. The incremental cost (less committed-use discount per vendor when traffic is split) is framed to Finance explicitly as the price of **availability and quality-jitter insurance**.

Sub-processor management under GDPR Article 28. Every model vendor above appears on the customer-facing sub-processor page. Sub-processor changes trigger the contractually agreed notification lead time (typically 30 days) before the platform routes any customer data through the new sub-processor. Customer opt-out (the **per-vendor veto**) is a tenant-level preference that the workspace broker enforces at routing time.

*Depth: **Trust, economics, and continuous adaptation** covers all six security layers, customer-data controls in operational detail, and the Autonomy Council mandate. Appendix P sources every claim to a specific study, contract term, or regulation.*

8. Peer examples worth knowing about

The platform argument does not claim novelty for every primitive; it claims coherent composition. Several peer organisations and industry patterns support specific components of the argument. Naming them makes the argument defensible against "nobody does this at scale" pushback.

Google SRE, toil framing (2016). The SRE book defined **toil** as the category of work that is manual, repetitive, automatable, tactical, devoid of enduring value, and grows at least as fast as the service. Ambient agents are the AI-native generalisation of the SRE toil argument extended to security, docs, dependencies, data, and product analytics. The signal-to-noise budget the platform imposes on each ambient agent is the direct analogue of an error budget applied to interrupt-worthy output.

Meta CodeCompose, FSE 2024. With 30,000+ engineers instrumented, Meta had to distinguish **suggestion acceptance** from **productivity** because a suggestion accepted and then rewritten is not the same as work saved. This is the direct empirical foundation for treating agent-authored share

as a **leading indicator only** and reserving outcome credit for DORA metrics on merged, verified changes.

Spotify Backstage. The originating example of an internal developer platform with paved roads exposed through a coherent gateway. The MCP + skills argument here is the AI-native generalisation: paved roads speak MCP rather than only HTML.

Amazon Bar Raiser. The rotating-reviewer discipline that keeps hiring standards from decaying under pressure. The Autonomy Council is the direct organisational analogue for autonomy progression — a body chartered to hold the standard against the pressure to grant tiers without evidence.

Databricks Unity Catalog. Governed data access exposed to compute regardless of engine. The MCP-mediated feature-store and model-registry pattern in the data-science section directly mirrors this — governed capability access exposed to agents regardless of runtime.

Netflix OpenRewrite. Programmatic large-scale refactoring across a fleet. The direct precedent for cross-repo sweep mechanics.

Anthropic and OpenAI Zero Data Retention offerings (2024-2025). Enterprise-tier contract terms explicitly designed for the customer-data-as-governed-egress framing. Their existence is what makes the customer-data-controls section shippable rather than aspirational.

None of these individually is what this document proposes. The proposal is the **composition** — a single platform serving production work, POCs, cross-repo sweeps, and data-science workflows under one governance and evidence discipline.

9. Making the product agent-ready — the engineering bill for reach

Being reachable from wherever a user starts work is the commercial argument. This is what it costs to be reachable safely.

The caller changed. It arrives with a token, a task it did not write, no patience and no eyes — it will not notice that a result looks odd, will not stop at the first page, and will combine what you return with material from three other products before anyone sees it. A permission model built for a person browsing a UI is being asked to hold under a different load.

Point authorization is not enough. Point asks whether a principal may act on a known resource — the question most product permission models were built for. **Set authorization** asks which resources satisfy both the query and the policy, and it governs titles, snippets, citations, facets, counts, ordering, and whether a continuation page exists. Those are observable outputs even when no document body is returned. A search endpoint is a security boundary and a retrieval service is a policy engine, whether or not anyone calls them that (R377, R378).

Four consequences follow, and each one is a place products fail quietly.

- **Post-filtering is safe and still wrong.** Global ranking D1–D6, the caller may read D2, D5, D6.

Fetch the top three, drop what is not permitted, return one result — when the authorized top three were D2, D5 and D6. No fixed overfetch factor fixes this across arbitrary permission distributions (R379). A final authoritative check removes false permits; it cannot recover what a stale prefilter excluded. **Revocation safety** and **grant freshness** are separate guarantees that fail in opposite directions.

- **Counts and facets disclose.** "Twenty investigations match this executive's name" leaks with every record still hidden. Exactness is a claim about a named corpus, predicate, and snapshot — adding "about" to a number removes precision, not disclosure. Bind cursors to the principal and the policy revision so pagination cannot restore access someone just lost.
- **Derived artifacts keep their dependencies.** Embeddings, extracted text, summaries, caches, and transcripts inherit their source's policy; a permissions-only change invalidates them while the bytes stay identical. For an inseparable synthesis, the audience is the intersection of recipients permitted for every source used.
- **Delegation is an intersection, never a union.** Sponsor rights ∩ application capabilities ∩ agent charter ∩ declared purpose ∩ resource policy, evaluated at execution against the exact target. Purpose comes from an approved workflow, not from model-generated text. A child agent narrows; it never accumulates. A disabled tool rejects direct invocation, because hiding it from discovery is not enforcement (R384, R385).

Cross-organization collaboration is the case tenant-shaped thinking misses. External counsel in a deal folder, a supplier in a sourcing workspace, a CRO on a sponsor's study — each owner domain stays authoritative for its own resources, link-based access never becomes a permanent searchable grant, and the destination audience for a synthesis is checked separately from the right to read its inputs.

What the integrator on the other side actually experiences is a set of promises. Publish a **capability ledger** separating announced from enabled from verified per tenant — an announced control is not a safeguard. Ship a conformance suite they can run against their own tenant, because correctness claims only you can verify are marketing. Give three worked examples rather than thirty endpoints: a delegated read, an unattended service workflow, and a consequential write, each showing its expected failures and audit fields. Version tool schemas like APIs with a deprecation window; the caller will not read a changelog. And meter what agents consume, attributed to the sponsoring principal rather than the integration's service account — seat pricing assumes a human ceiling on consumption that agents remove.

*Depth: the full document develops this in **Making a product agent-ready**, with the measurement set in Appendix T Category I.*

10. A first-6-months sequence

Any plan that treats the maturity journey as a launch fails on week 3. What follows is a **sequence** — an opinionated ordering of the first six months of platform investment that maximises the 90-day defensibility of the programme while setting up the compound value of the next two years.

Month 1 — Foundations audit + tier map v0. Score every repository against the agentic-readiness ladder. Publish the histogram. Score the top 5 candidate agent capabilities against the maturity matrix. Publish the tier map v0. Stand up the Autonomy Council with an initial membership and a charter. This month produces no automated capability; it produces the **shared vocabulary** the rest

of the programme runs on.

Months 2–3 — MCP gateway pilot + skills registry v0. Ship the MCP gateway with 3–5 initial skills wrapping the most-touched paved roads (auth wrapper, tenant scoping, schema check, primary build system, primary deploy path). Wire the decision-packet system into the existing PR review flow for one pilot team. Ship the workspace broker with vendor-hosted placement for T0/T1 work.

Months 4–5 — First sweep + first auto-approval class. Execute one CVE-class sweep end to end (row 1 + row 6 corner of the matrix). Progress the CVE bump task class from T1 to T2 based on measured evidence from the pilot team. Ship the comprehension record on the pilot team's decision packets and measure the reviewer self-report gap for a baseline.

Month 6 — Autonomy Council reauthorization cycle 1. First quarterly review. Publish the tier map v1 with the movements earned in months 2–5. Publish the debt trajectory for the pilot team's systems. Present the first quarterly evidence pack to the CFO and the CISO. Commit to the next six months' journey plan based on measured evidence rather than roadmap ambition.

This sequence is deliberately **sub-linear** in agent capabilities shipped in the first quarter. The compound value shows up in quarters 3–8 as the tier map, the readiness histogram, the debt trajectory, and the Autonomy Council operating rhythm all compound. A plan that ships 15 agent capabilities in the first quarter and then discovers it has no vocabulary for scoping, governing, or measuring them has **made the problem harder**, not easier.

11. Debt is first-class engineering

Speed without understanding is technical debt. The industry conversation is almost entirely about generation velocity and almost never about the debt that arrives with it. This platform treats debt as a first-class engineering concern — with mechanisms and measurements — rather than externalising it onto the customer organisation.

Three debts, not one. The fixes differ.

- **Technical debt** (Cunningham 1992) — the classic. Code smells, aged dependencies, brittle tests, ADRs whose rationale has been forgotten. This is what static analysis and dependency graphs measure. AI accelerates its accumulation because generation outpaces refactoring.
- **Comprehension debt** (Osmani 2026) — the individual reviewer layer. The gap between how much code is in the system and how much of it any human genuinely understands. Grows invisibly because "the tests pass" masks it. Measurable via comprehension quizzes, review-latency, and reviewer override patterns.
- **Cognitive debt** (Storey 2026, drawing on Naur's **Programming as Theory Building**) — the shared team-level layer. The lost common mental model of the system. A programme is a theory in the developers' heads, not a text file; when the shared theory decays, the team's ability to safely change the system decays with it. Cognitive debt is what makes a team hit a wall at week seven or eight of AI-accelerated work.

Technical debt is what static analysis sees. Comprehension debt is what a single reviewer notices when they try to explain a change and can't. Cognitive debt is what an on-call engineer discovers at 3am when the person who "wrote" (delegated) the code is not on the team and neither is anyone

else who understands the intent.

Platform mechanisms:

- **Comprehension record** on every T2+ decision packet — evidence, not a checkbox. Reviewer names: what changed, why, what risk class, what tests they trust. Falsifiable at postmortem.
- **Debt delta** on every change — machine-readable per-change delta: added lines, complexity delta, test-coverage delta, dependency delta, doc-freshness delta. Aggregated per system per week.
- **Autonomy Council debt trajectory review** — per system, not per PR. Trending in the wrong direction triggers tier regression until fixed.
- **Retirement as a first-class action** — dead skills, dead constitutions, dead systems all get cleaned up on cadence.

The five landmark numbers. Each is developed in the full document and sourced in Appendix P.

Table 1. The landmark numbers

Number	Claim
22.7% persistent	Liu et al. 2026: across 302,600 verified AI-authored commits in 6,299 GitHub repositories, 22.7% of AI-introduced issues are still present in the repository's latest version. **The debt persists.**
+24% felt / −19% measured	METR July 2025: 16 experienced OSS maintainers on their own codebases **felt** AI made them ~24% faster; measured result was ~19% **slower**. The gap itself is evidence of comprehension debt.
17% gap; 40% vs 65%	Anthropic RCT (52 engineers): AI-assisted −17% on comprehension. Passive-delegation users <40%. Inquiry-driven users >65%. **Same tool, different discipline.**
25% adoption → −1.5% / −7.2%	DORA generative-AI analysis: 25% AI adoption increase associated with 1.5% throughput and 7.2% stability . Counterweight to unsupported velocity claims.
20–40%	McKinsey 2020: technical debt = 20–40% of a company's technology-estate value. The baseline the AI-native platform inherits.

The single number to put in front of any skeptic is the METR gap: +24% felt, -19% measured, on experienced engineers doing work on their own codebases with a tool they had access to and knew how to use. The tool alone does not determine the outcome. The discipline does.

*Depth: **Debt, comprehension, and continuous understanding** covers the three debts, all six platform mechanisms, and the full sourcing chain.*

12. Acceleration, safe experimentation, and data science

Same platform primitives that make production work governed also make cheap experiments

possible.

Ephemeral agent workspaces for POCs. The same workspace broker that routes production work spins up sandboxed T0 or T1 spaces for spikes, proofs of concept, and one-off analyses. Cleaned up on schedule. No dedicated experimentation platform to build and staff separately.

Skills reuse across surfaces. The auth wrapper, tenant scoping, and schema check are the same in production code and in a Jupyter cell. POCs get enterprise-grade primitives on day one rather than being rebuilt to production quality if they survive.

Cheap experiments at T0. Read-only exploration, no writes, no auth escalation. Cost per experiment drops toward zero; the number of POCs the organisation can afford in a quarter goes up by an order of magnitude.

Evidence chain applies to prototypes. Even throwaway spikes carry provenance, so the ones worth promoting to T2+ arrive with their own audit trail — no rewrite required to make them governed.

Cross-stack fluency — feature and concern. Agentic development lowers the tech-stack boundary. Backend engineers can contribute plausible UI changes; platform engineers can spin up Python pipelines for POCs; teams can evaluate unfamiliar SDKs without a full onboarding cycle. This is **real workforce flexibility**. It is also a **shallow-familiar** comprehension risk — the cost of producing work in an unfamiliar stack drops faster than the cost of evaluating it. Named and treated as a first-class concern with routing (consequential cross-stack changes route to a deep-familiar reviewer) and measurement (cross-stack contribution share paired with the shallow-familiar rate).

Data science on the same platform. Honest disclosure: the full document does not have a dedicated DS chapter. The platform primitives apply directly:

- Workspaces for notebook and training runs — same broker, same trust-boundary placement rules.
- Feature stores and model registries as MCP consumers — one gateway, not a bolt-on integration.
- Ambient data / ML agents — data-quality checks, drift detection, retraining triggers — governed by the same Autonomy Council on a signal-to-noise budget.
- Evidence chain applies to model artefacts — provenance, version pins, dataset hashes travel with every promotion.
- **Upshot.** DS is not a second platform. It is one more consumer of the same one.

*Depth: **Organisational and enterprise product implications** covers cross-stack fluency in operational detail. **Autonomous and ambient agents** covers the ambient data / ML lifecycle patterns with worked exemplars in Appendix R.*

13. Ambient agents — the majority of the opportunity

Reactive AI (feature requests, bug fixes, PR review) reduces the **per-item cost** of visible work. **Ambient AI reduces the volume of toil that visible work is dragged down by.** The two together

produce the compounding effect. An organisation that only invests in reactive agents ships faster but keeps accumulating the toil that made it slow to begin with.

Every ambient agent has three axes: trigger (scheduled, signal, event, continuous, curiosity), action (T0 through T4 on the autonomy ladder), and termination (idle, checkpointed, delegating, waiting for human decision). **Governance attaches per-axis.**

Eight consequential classes, each with a worked exemplar in the full document's Appendix R:

- **Repository hygiene** — dead code detection, dependency drift monitoring, doc freshness scoring, retirement candidates. The archetypal ambient class; the most under-invested in most organisations because no team owns it.
- **Fleet-wide refactoring and migration** — coordinated code changes at scale. The mechanism for making cross-repo sweeps practical.
- **Observability + FinOps + infrastructure optimisation** — cost anomaly detection, resource-waste identification, alert- calibration drift, dashboard pruning.
- **Continuous security and compliance** — vulnerability scanning, posture drift, sub-processor list currency, secret-scanning false-positive triage.
- **Knowledge production and organisational learning** — auto- generated docs, ADR summaries, runbook updates from incident patterns, retrospective-to-action conversion.
- **Ecosystem and market intelligence** — vendor scanning, framework tracking, licence-drift detection, competitor release monitoring. The CVE feed that starts a sweep is a member of this class.
- **Data / ML lifecycle** — dataset drift, model-performance monitoring, retraining triggers, feature-store freshness. The primary vehicle for the data-science-on-the-same-platform argument.
- **Product / customer analytics** — usage patterns, adoption signals, feedback classification, feature-flag graveyard monitoring.

The failure mode to design against: the silent army. Dozens of scripts-with-a-personality running on cron, accumulating outside any owner's view, until one of them exceeds its budget or blast radius and no one is on call for it. Prevented by **standing service identity** per agent, **portfolio inventory** owned by an ambient-agent PM, **signal-to-noise budget** per agent, and **quarterly Autonomy Council reauthorization**.

*Depth: **Autonomous and ambient agents** covers all eight classes with implementation fields. Appendix R is the reference catalogue of 20+ worked exemplars.*

14. Organisational implications

The platform is not only a technical proposal. It reshapes roles, career ladders, hiring, and change management. Naming these implications up front, and building the roles the platform requires, is what turns the technical argument into an organisational outcome.

Roles become more permeable. Product, design, engineering, architecture, QA, security, and operations share more of a common surface. Deep expertise moves toward higher-consequence

decisions rather than disappearing. The platform must serve everyone who participates in software creation, not only a person typing code. This is a talent-ladder change more than a technology change: the organisation is now hiring for judgment, platform contribution, and evidence quality rather than for pure throughput.

Two new roles emerge from the ambient surface. The **ambient-agent product manager** owns the ambient portfolio — roadmap, constitutions, signal-to-noise budgets, sunset discipline. The **agent SRE** is on-call for the ambient agents themselves — runbooks, runaway response, feedback-loop detection, cost anomalies. Neither is a rebrand of an existing role; both need explicit staffing and reporting-line decisions that most organisations have not yet made.

Enterprise products must participate in agentic workflows. AI features inside an application are not enough. Products must expose **trusted capabilities** discoverable and composable by agents on the customer's chosen surface. Content-safe capabilities rather than generic CRUD, tiered by reversibility (T1 reads → T2 reversible → T3 consequential). A product without an agent-safe capability surface is unbuildable-on by agents in governed environments, which increasingly means it is unbuildable-on by anyone at enterprise scale.

Change management is a platform product. Always-on, with enablement, evaluation, alpha cohorts, and vendor scanning as persistent workstreams. Not a one-time adoption programme.

Cross-stack fluency — a feature and a concern. Alongside role permeability, agentic development lowers a second boundary: the **tech-stack silo**. A backend engineer with agent assistance can contribute a plausible React component to a UI they would previously not have opened. A platform engineer can spin up a Python data pipeline for a POC in an afternoon. A team can evaluate a new SDK — a database driver, an observability client, a payments library — in a language they do not maintain, without a full onboarding cycle.

As a feature: workforce flexibility of a kind the industry has not previously had. Cross-stack contribution reduces the coordination tax on small changes that would otherwise stall waiting for the "right" team. POCs that were non-starters — because the cost of a senior engineer learning enough of an unfamiliar SDK to evaluate it exceeded the value of the evaluation — now become tractable.

As a concern: the cost of **producing** work in an unfamiliar stack drops faster than the cost of **evaluating** it. The failure mode is **shallow-familiar**: confident enough to merge, not deep enough to catch what should not have been merged. Named as a first-class concern in the full document, with routing (consequential cross-stack changes route to a deep-familiar reviewer) and measurement (cross-stack contribution share paired with the shallow-familiar rate).

Deskilling risk. If the org optimises for delegated throughput without the discipline of the comprehension record and the Autonomy Council debt review, engineers stop building deep familiarity with the systems they own. The Anthropic RCT result — passive-delegation users at <40% comprehension, inquiry-driven users at >65% on the same tool — is the empirical anchor for why **discipline determines the outcome**, not the tool.

*Depth: **Organisational and enterprise product implications** covers the two new roles, the enterprise product participation argument, cross-stack fluency, and the change-management-as- platform-product argument in operational detail.*

15. Positioning against the vendor market

The vendor market has largely converged on the **orchestration framing** — agents move from completion to orchestration, the developer's job is to direct outcomes. Cursor states this bluntly: **"most code will be written by AI agents and the developer's job is to orchestrate them."** Devin delivers it as end-to-end autonomous engineering with multi-agent orchestration and a self-healing codebase on the roadmap. GitHub Copilot ships it as Agent Mode plus a Coding Agent that turns issues into PRs in the background. Anthropic Claude Code exposes it as `/goal` with success criteria plus worktrees for isolated branch work. Sourcegraph Amp positions itself as a purpose-built agent that loops until a job is done. Augment delivers it as risk-tiered auto-approval on retrieval-augmented context. That debate is effectively over.

The market **diverges on the how.** Augment and Devin place more of the organisational scaffolding **inside the product** (retrieval- augmented context, risk-tiered auto-approval, journey framing). Cursor, Copilot, Claude Code, and Amp explicitly leave the organisational layer to the customer enterprise. **This document is one attempt to fill the gap the second group leaves.**

Two systematic silences in the surveyed vendor material:

- **Silence on debt and cognitive load.** No surveyed vendor engages the METR gap or Liu persistence data publicly. Velocity metrics dominate.
- **Silence on customer-data-as-governed-egress.** Every vendor offers enterprise-tier contract terms. None exposes per-request tenant-queryable audit as an architectural primitive.

Practical implication. The vendor market can be used to **fund the primitives** — workspaces, skills registries, gateway-mediated tool calls, ambient triggers, goal-directed autonomy — while the **governance, journey management, and debt accounting remain organisational work**.

Depth: Appendix S surveys six vendors (Cognition/Devin, Cursor, Augment, GitHub Copilot, Anthropic Claude Code, Sourcegraph Amp) with position, claims, agreements, and silences per vendor.

16. The eight decisions this proposal would defend hardest

Eight architectural commitments the platform depends on. Each cuts off a shortcut that looks faster this quarter and costs badly next year.

1. **The MCP gateway is the only path to enterprise systems.** Agents call through it or they do not call at all.
2. **The decision packet is a first-class platform object.** Not an inbox filter, not a Slack notification — a durable, evidence-bearing work object.
3. **Autonomy is per-task-class, earned on evidence.** Never granted globally, never assumed permanent.
4. **Model routing is abstracted behind the gateway.** Single-vendor optionality preserved as a business asset.

5. **Security is an independently enforced control plane** above the agent runtime. A compromised agent cannot disable its own guardrails.
6. **Customer-data flow is a governed, audited egress event**, not an API call detail.
7. **The ambient portfolio has an inventory, a council, and a budget.** Or it becomes the silent army.
8. **Specifications carry provenance.** Signal identifiers, sample size, and a validation state on the requirement record — and no autonomy above T3 on problem selection.

If a reader accepts only decisions 1 and 2 — MCP as the single path and decision packet as the primitive — they still have most of the value. The other five are how the organisation avoids accumulating debt that undoes those first two.

17. Measurement in one page

Categories the platform tracks, paired with counter-metrics that keep the velocity signals honest.

Category	Anchor metrics + counter-metric
DORA baseline	Deployment frequency, lead time, change-failure rate, MTTR. *Never publish deployment frequency without change-failure rate.*
Attention economy	Human-attention-per-verified-outcome by matrix cell (the meter). Decision-packet volume, deduplication rate, attention-per-packet. *Counter: escalation rate.*
Cognitive & comprehension	Comprehension-record completion rate, reviewer self-report gap, METR-gap replication on the org's own engineers. *Counter: completion rate at 100% with widening self-report gap = process theatre.*
Debt trajectory	Debt delta per change, debt trajectory per system, AI-authored persistence rate, rework rate. *Counter: agent-authored share rising while debt trajectory worsens = externalised debt.*
Autonomy movement	Cells advanced per quarter, coverage share, time-to-full- auto-approval at row 1, auto-approval blast radius. *Counter: cells advancing while coverage share stagnates = capabilities nobody uses.*
Ambient + sweep	Portfolio inventory, signal-to-noise adherence, reauthorization completion, sweep throughput, coordination-hour reduction. *Counter: sweep throughput rising with failure rate rising = executing faster than readiness supports.*
Repository readiness	Repository distribution across tiers, tier lift per quarter, Tier-2 investment. *Counter: tier lift with no matrix coverage gain = ladder work not paying off.*

Anti-metrics — never optimise for in isolation: total agent invocations · lines authored by agents · time from ticket-open to ticket-closed · reviewer-approved PR count · automation coverage percentage · agent notification count. Every one of these can be gamed to look positive on partial data while the underlying system degrades.

The quarterly review composition — seven slides. The Autonomy Council review is not a set of disconnected dashboards; it is one composite story. Slide 1: **the meter** — human-attention-per-verified-outcome by matrix cell, quarter over quarter. Slide 2: **velocity paired with quality** — DORA metrics paired with change-failure rate and MTTR. Slide 3: **the debt counter-signal** — debt trajectory per system plus METR-gap replication; if either worsens while Slides 1 and 2 improve, the platform is externalising debt. Slide 4: **journey progress** — cells advanced and coverage share against the 12-month journey plan. Slide 5: **ambient and sweep economics** — portfolio health plus sweep throughput and coordination-hour reduction. Slide 6: **readiness** — repository tier histogram. Slide 7: **one decision** — the Council names one adjustment to policy, portfolio, or investment coming out of the quarter. The seven-slide format is the point: longer reviews attract optimisation of the review itself; shorter reviews cannot represent the composition.

Depth: Appendix T (Measurement framework) defines every metric above with cadence, leading-vs-lagging classification, related mechanisms, and per-chapter 'How we measure this' subsections tying each metric back to the section that owns it.

18. Regulatory posture

Three regulatory realities shape the platform's architectural inputs from day one. None of them is optional.

Data residency is a set of jurisdictional obligations — GDPR in the EU, UK GDPR, PIPEDA in Canada, LGPD in Brazil, PDPA in Singapore, and roughly two dozen state-level regimes in the US — that constrain where customer data may be processed. The workspace-broker placement decision is a residency decision: a tenant with EU data has its agent workspaces placed in the EU region and its model-vendor routes constrained to EU-region-compliant deployments.

The EU AI Act (in effect 2025-2026) is the first cross- jurisdictional risk-tiered regime for AI systems. Enterprise- scoped agentic software development lands mostly outside the Act's **high-risk** categories, but agents that touch employment, credit, education, law enforcement, or critical infrastructure inherit high-risk obligations — transparency, human oversight, data governance, technical documentation, post-market monitoring, and (for general-purpose AI systems) systemic-risk evaluation. The platform's agent constitutions, evidence chain, comprehension records, and Autonomy Council governance are what make high-risk classification defensible: they are the technical documentation the Act requires.

Sector-specific regimes — HIPAA in US healthcare, FedRAMP for US federal, MAS TRM in Singapore financial services, SOX for public companies, PCI-DSS for card data. The platform does not attempt to solve any of these; it provides the primitives (governed egress, per-request audit, workspace placement, constitution enforcement) on which sector-specific certifications can be assembled. The full document is explicit that the SOC 2 / ISO 27001 / FedRAMP mapping is the customer team's job; this platform provides the substrate.

The failure mode. Enterprises that treat regulation as a compliance-team problem after the platform ships end up retrofitting mechanisms — residency, per-request audit, sub-processor management — into a runtime that was not designed for them. The retrofit is expensive, incomplete, and often requires re-issuing customer contracts. The platform proposed here treats these as **architectural inputs from day one**, which is why the workspace broker, the MCP gateway, the constitution mechanism, and the Autonomy Council all appear in the earliest tier-map decisions rather than in a compliance appendix.

19. What this proposal does not cover

Honest limits, so the reader does not arrive expecting the wrong artefact.

- **Not a vendor evaluation.** Reference architectures and patterns, not a Copilot-vs-Cursor-vs-Codex-vs-Claude-Code ranking. The full document's Appendix Q is a landscape scan; Appendix S surveys vendor **positions** against this document's thesis, not vendor **ratings**.
- **Not sized to a specific enterprise.** Ratios and evidence patterns, not a line-item budget. A CFO wanting a number needs to feed the ratios through their own headcount, model spend, and adoption curve.
- **Assumes baseline platform maturity.** Without paved roads, a service catalogue, working CI/CD, and workload identity, the MCP-mediated platform reads as science fiction. Start with the foundations, then this becomes tractable.
- **Multi-service pre-deployment testing is unsolved.** Named as an open question in the full document rather than pretended otherwise. Any team promising row-5 or row-6 automation without a credible answer here is either automating something smaller than they claim or accepting a risk profile the organisation should audit.
- **Does not prescribe org design or reporting lines.** Names roles and permeability. Leaves the org chart to you.
- **Does not do the SOC 2 / ISO 27001 / FedRAMP mappings for your environment.** The framework is present; the mapping is your team's job.
- **Data-science chapter is thin.** The platform primitives serve DS directly (§ Acceleration, safe experimentation, and data science). A dedicated DS chapter would need its own document.
- **Does not prescribe a product-discovery methodology.** The requirements and discovery section makes discovery evidence-bearing and puts it on the autonomy ladder. It does not adjudicate between Cagan, Torres, jobs-to-be-done, or whatever the product organisation already runs.

20. Where to go deeper

The full document contains everything this edition skipped: the reference operating models (Part Five), the appendix-length treatments of MCP focus, workspace placement, PR review, verification chains, A2A component models, stakeholder decisions, consolidated sourcing, the ambient-agent catalogue, and the measurement framework in full detail. The abridged edition you are holding preserves the **argument**; the full document preserves the **audit trail**.

Artefact	URL
Full document (~385 pages, neutral)	geekychris.github.io/ai-native-sdlc
Full document (case-study variant)	/case-study.html

Artefact	URL
Concise edition (~230 pages, every chapter, trimmed)	/concise.pdf
Cliff-notes companion (~13 pages, skim in 15 min)	/companion.pdf
Hierarchical punchlist (~40 pages, ~370 tasks)	/punchlist.pdf
Interactive citymap of the document structure	/citymap.html
This abridged edition	/abridged.html
Latest release (all artefacts)	github.com/geekychris/ai-native-sdlc/releases/latest
Source repository	github.com/geekychris/ai-native-sdlc

Every element on every page of this edition has a chapter or appendix behind it in the full document. If a claim reads counter-intuitive, the audit trail is one click away.