AI Native Developer Platform — Cliff notes

For readers deciding whether to invest a day in the full document.

Companion to **AI Native Developer Platform: Reimagining the software development lifecycle for agents, evidence, and governed action**.

Full document (~390 pages): geekychris.github.io/ai-native-sdlc

+ Concise edition (~245 pages): /concise.pdf · Abridged edition (~20 pages): /abridged.pdf

+ Latest release (all artifacts): github.com/geekychris/ai-native-sdlc/releases/latest

---

This companion exists so a Principal Engineer, staff engineer, or CTO can decide in fifteen minutes whether the full ~390-page document repays a day of reading. It does not replace the full document; every claim below is developed, sourced, and qualified there.

The vocabulary is the same as the main document — **decision packet**, **agent workspace**, **execution substrate**, **autonomy tier T0–T4**, **agent constitution**, **ambient agent**, **comprehension debt**, **cognitive debt**, **intent debt**, **soft rail**, **hard rail**, **signal**, **opportunity**, **evidence chain**, **maturity matrix**, **agentic readiness tier**, **cross-repo sweep**. Full definitions are in the main document's glossary chapter, which also carries a complete acronym table.

Enough of it to read this on its own:

Term	In one line
MCP	Model Context Protocol — the governed boundary through which an agent discovers and invokes tools. A capability boundary, not a control plane
A2A	Agent-to-agent — collaboration between independently operated agents, where MCP is agent-to-tool
SDLC	Software development lifecycle, used here for the whole loop from customer signal to production learning, not only the coding part
Decision packet	The durable unit of human judgment: question, owner, evidence, options, recommendation, blast radius, rollback, and a resume point
Autonomy tier	T0 observe, T1 recommend, T2 reversible action, T3 bounded action, T4 autonomous in class. Earned per task class on measured evidence, never granted globally
Agent constitution	Versioned, signed, runtime-enforced policy: what an agent may do, must do, must never do, and when it escalates
Soft rail	A capability an agent loads at the point of work — a skill, a review command, a scaffold. Adopted, not enforced

Term	In one line
Hard rail	A bound on what an agent may do, enforced regardless of cooperation — the gateway, the tier, the constitution
Evidence chain	Signal, opportunity, specification, work item, change set, decision packet, release, operational event, learning — with learning writing back
Comprehension debt	The gap between how much code is in the system and how much of it any human genuinely understands
ZDR	Zero data retention — a vendor commitment to keep no copy of what you send
DORA	DevOps Research and Assessment, the source of the four delivery keys
METR	Model Evaluation and Threat Research, whose study found developers ~19% slower while feeling ~24% faster

Contents. (1) The claim in one sentence. (2) The ten principles. (3) The maturity journey. (4) Twelve pitch points that follow from the claim. (5) Who reads what. (6) Where this differs from vendor pitches. (7) Landmark numbers. (8) What is still being argued about. (9) What this document does not cover. (10) Where to find things.

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 organization 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.

Everything else in this document — the SDLC transformation, the MCP-mediated foundations, the governance and customer-data posture, the debt accounting, the acceleration and experimentation story, the data-science extension, the sweep orchestration, the measurement framework — is a consequence of that single claim. Read the rest as a series of specific engineering choices that only make sense if you already accept it.

2. The ten principles

The main document states these as invariants in its own chapter, before any mechanism. Everything after is an implementation of them, and a platform decision that breaks one is a decision to reopen the chapter.

1. **Asynchronous by default.** Delegation is the normal mode. Nobody sits watching an agent type, and the concurrency comes from the engineer not being inside each execution.
2. **Interactive on demand.** Interactivity is the escalation path — when a UI has to be felt, when a root cause will not resolve from a trace, when an ambiguity will not settle.

3. **Abstraction is escapable.** Decision packet down to logs, shell, running preview, attached debugger, and back out. A senior engineer will reject a platform that offers no way down.
4. **Workspaces are shareable.** A running work context is handed to a colleague under differentiated access, so people collaborate on the artifact rather than a description of it.
5. **Workspaces are resumable and portable.** A workspace is a logical package of work context, not a process — suspended, resumed, handed on, reconstituted elsewhere.
6. **Execution is heterogeneous.** Linux, macOS and Windows workers, emulators, device pools, and the engineer's own laptop are all legitimate substrates.
7. **Execution follows the work.** Placement is a scheduling decision made from toolchain, data class, residency, latency, hardware, and cost — not a fact the engineer carries around. **Remote is one placement, not the target state.**
8. **Evidence is discipline-specific.** Backend proves itself with contracts, UI with visual diffs and a device matrix, data and ML with provenance and drift. Autonomy ceilings follow the strength of the evidence a discipline can produce.
9. **Capabilities are governed.** Agents get capabilities, not credentials. Authority is delegated, bounded, observable, revocable.
10. **Human judgment stays the scarce resource.** Everything above serves concentrating attention on the decisions that need it.

Read the principles chapter first if you read nothing else. It is the shortest chapter in the document and the one the rest answers to.

3. The maturity journey — how you get there

The transformation is **iterative**, not a launch. Two dimensions of maturity move at different speeds: **complexity of the change** (from a CVE bump to a multi-service feature with UI) and **breadth of the change** (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.

The far target — spec for a large multi-service feature planned and executed to deployment with minimum human interaction — is a north star, not a claim of what is shippable next year. What **is** shippable now is the **routine and mechanical slice** where most engineering hours already go. Multiple empirical studies place that slice in the 70–85% range of engineering activity. **Covering the 80/20 case well is a larger prize than solving the far target poorly.**

The main document introduces four artefacts to make the journey governable:

- **The maturity matrix.** Complexity bands (rows) × SDLC phases (columns). Cell shading = the autonomy tier viable **today**. The Autonomy Council owns the matrix and publishes a **tier map** each quarter with the evidence supporting each movement.
- **The 12-month journey plan.** Which cells the platform intends to advance next, in what order, on what evidence.
- **The agentic-readiness ladder.** Five tiers (Tier 0 agent-hostile → Tier 4 agent-native, with a

multi-repo change harness). Each tier unlocks the next matrix band. **You cannot skip tiers.**

- **Cross-repo sweep orchestration.** The broad-simple corner of the matrix, treated as a first-class capability rather than a series of independent per-repo changes.

the maturity-journey chapter of the main document develops all four. Auto-approval at the trivial band, and the open question of multi-service pre-deployment testing, are both named explicitly.

4. The twelve pitch points

Each pitch point below follows from the claim and from the principles above. Roughly a page each.

4.1. 1. SDLC transformation — engineer as orchestrator

- Engineers move from executing every step to **orchestrating outcomes**: define intent, constraints, tests, SLOs, failure policy, authority; delegate bounded work to agents; intervene on the consequential slice.
- 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.**
- The central primitive is not code. It is the **decision packet** — a durable, deduplicated, evidence-bearing work object with an owner, a deadline, a recommendation, a blast radius, and a rollback plan.
- Autonomy is on a **T0–T4 ladder**, earned per task class on measured evidence. **Never granted globally.**
- Review, testing, and operations become parallel agentic disciplines with **progressive operational authority** on the same ladder.

Read Part One + the maturity-journey chapter when you are trying to decide what "AI-native SDLC" means in operational terms and how to sequence the rollout.

4.2. 2. Enterprise resource connection — MCP + skills + A2A

- Organisational knowledge — SDKs, 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, permissions, tests, and retirement.
- **Paved roads must be agent-accessible** through the same governed interfaces humans use. Otherwise agents silently reinvent them.
- **MCP and REST change the agent inner loop.** The platform ships an MCP-gateway model mediating auth, audit, rate limits, schema stability, and least-privilege scoping. "Give the agent a tool" is not "give the agent the keys to production."

- **A2A** extends the loop across specialist agents (test generator, security reviewer, migration planner) under a coordinator that keeps the decision packet clean.
- **Agent orchestration and runtime context assembly** are first-class platform products. Orchestration is the **when**: a unified trigger substrate (event router / scheduler / signal engine / continuous runtime / curiosity budget) that closes off bespoke cron and webhook plumbing. Context assembly is the **with-what**: permission-aware retrieval, context minimization, derived-representation policy inheritance, and constitution injection at runtime. Retrieval is part of authorization (never filter only the final answer); prompt injection is an authorization problem (retrieved instructions are untrusted, and tool policy lives outside model persuasion).
- **Managed context repositories** — code, docs, ADRs, runbooks, tickets, incidents, skills, constitutions — each with an owner, schema, freshness contract, retrieval interface, provenance chain, and retirement path.
- **Agent-ready project specification** — an opinionated project layout (runtime config, sandbox spec, tests, docs, dep manifest, rollback contract, owner, constitution binding) that every repository is scored against. The twelve-factor-app analogue for the agent era.
- **Ten properties of an agent fit for corporate use** — a single checklist a security review or procurement process can use directly: standing service identity, least-privilege delegation, execution isolation, egress control, tool policy that separates read from write, retrieval-as-authorization, the derived- representation policy tail, a comprehension record on T2+ changes, per-request outbound audit, and quarterly reauthorization. An agent missing any one of the ten is not ready to point at production or customer data.
- **Local and cloud MCPs are different threat models behind one gateway.** Local MCPs (stdio, workspace-scoped — filesystem, git, sqlite, browser drivers) fail open on the developer's machine; cloud MCPs (HTTP+SSE to enterprise systems — source control, ticketing, docs, paging, chat) carry tenant data across a network boundary. The gateway is where both are inventoried, scoped, and audited; the transport is not the control.
- **Ad-hoc tools become MCPs, not scripts.** The one-off script an engineer writes to query a system becomes a registered, owned, permission-scoped MCP tool. That is how the tool surface stays governable instead of accreting shadow automation.
- **Two families of capability, and the internal one matters more.** What the organisation buys — source control, work management, observability, security, CI — is largely a procurement question. What it builds is what encodes how **this** organisation engineers software: golden examples, ownership and escalation, approved SDKs, compliant service creation, workspaces, test data, canary, runbooks. Golden examples show the boundary is not only about actions; returning the approved implementation of a pattern is supplying authorised context.
- **Expose governed outcomes, not application surfaces.** A hundred and fifty REST endpoints should not become a hundred and fifty tools. `get_work_context`, `record_decision`, `create_agent_workspace` — named for what the organisation supports, not for the vendor's data model. Selection accuracy collapses as a catalogue grows.
- **MCP exposes capability; it does not grant authority.** A capability boundary looks like a control plane and is not one. Governance mediates the path between orchestrator and capability rather than forming a perimeter — a ring drawn around the ecosystem still lets anything inside it call anything else. MCP is a boundary, not the architecture, and **MCP server count is a poor maturity metric.**

Read Part Two when you are scoping the internal-platform investment required to make agentic work productive. Read §Properties of an agent for corporate use on its own when you are the security reviewer.

4.3. 3. Governance + customer-data privacy

- **Every prompt is a governed data-egress event, not an opaque API call.** Six operational controls: redaction on outbound, Zero-Data- Retention contracts (Anthropic ZDR, OpenAI Enterprise, Vertex, Bedrock, Azure OpenAI), data-class permission sets, sub-processor management under GDPR Article 28, per-request outbound-flow audit (tenant-queryable for subject-access), customer preferences with per-vendor veto.
- **Agent security is an independently enforced control plane.** Six layers — identity, authorization, execution isolation, egress control, evidence/audit, secrets — enforced **outside** the agent runtime so a compromised agent cannot disable its own guardrails.
- **Data residency, model IP, and regulation are first-order architectural constraints** — residency, indemnification, training-data provenance, and the EU AI Act determine which agents run where, on which data, with what retention posture.
- **Multi-vendor as an active routing strategy, not just contract fallback.** Frontier-model quality wavers day to day and frontier- vendor availability is not enterprise-tier. The model router treats the qualified vendor set as a **pool with health**, grades a small fraction of production requests in-flight against a per-capability evaluation harness, and rebalances weights on quality + health + cost + quota signals. Failover is a routing event, not a customer-visible outage. The incremental cost is the price of availability + quality-jitter insurance.
- **Agent constitutions** are a first-class runtime artefact — the declarative document specifying what an agent may do, must do, and must not do, with escalation, sunset, and reauthorisation rules. Enforced by the runtime, not by hope. Distinct from Anthropic's **Constitutional AI** training methodology.
- **Autonomy Council** reauthorises per system, per quarter — the governance body that owns constitutions, tier map, and the debt trajectory.

Read Part Three when you are pricing the platform to a CFO, answering a security or compliance review, or defending customer-data posture under a subject-access request.

4.4. 4. Debt is first-class engineering

- **Speed without understanding is technical debt.** AI accelerates its accumulation.
- Three debts, not one:
  - **Technical debt** — the classic.
  - **Comprehension debt** — one reviewer who cannot explain the change.
  - **Cognitive debt** — the team with no shared theory of the system.
- Platform mechanisms:
  - **Comprehension record** on every T2+ decision packet — evidence, not a checkbox.
  - **Debt delta** — every change reports the debt it added or paid down.

    - **Autonomy Council** reviews debt trajectory per system, not per PR.
- **The one number to keep in mind.** METR 2025: 16 experienced OSS maintainers on their own codebases **felt** AI made them ~24% faster; measured result was ~19% **slower**. The gap itself is the evidence of comprehension debt. Same tool, different discipline.

Read the debt chapter when a stakeholder is skeptical that AI-generated code carries long-term debt cost, or when you need to defend the comprehension-record mechanism against process-overhead pushback.

4.5. 5. Acceleration + safe experimentation and POCs

- Same platform primitives make production work **governed** and experiments **cheap**.
- **Ephemeral agent workspaces** — the workspace broker that routes production work spins up sandboxed T0/T1 spaces for spikes, POCs, one-off analyses. Cleaned up on schedule.
- **Skills reuse across surfaces** — the auth wrapper, tenant scoping, and schema check are the same in prod code and in a Jupyter cell. POCs get enterprise-grade primitives on day one.
- **Cheap experiments at T0** — read-only exploration, no writes, no auth escalation. Cost per experiment drops; count goes up.
- **Evidence chain applies to prototypes** — even throwaway spikes carry provenance, so the ones worth promoting to T2+ arrive with their own audit trail.
- **Cross-stack fluency (feature and concern).** Agentic development lowers the boundary between tech stacks — a backend engineer can contribute a plausible UI change, a platform engineer can spin up a Python spike in an afternoon, a team can evaluate an unfamiliar SDK without a full onboarding cycle. This is real workforce flexibility. It is also a specific **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 in the organizational chapter §Unlocking skill and stack silos.

Read the maturity-journey chapter + Part Four + Part Five when your organisation is torn between "we need to be careful" and "we need to move faster" — the same platform answers both.

4.6. 6. Data science on the same platform

- The full document does not have a dedicated data-science chapter, but the platform primitives apply directly.
- **Workspaces for notebook and training runs** — the same workspace broker places heavy compute against the trust boundary (customer-VPC for regulated data, restricted pools for cross-tenant work).
- **Feature stores and model registries behind the MCP gateway** — DS work is one MCP consumer among many, 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, same as with code.

- **Upshot.** DS is not a second platform. It is one more consumer of the same one.

Read Part Two + Part Six + Appendix J when your DS/ML organisation is being handed a separate platform investment; use this to argue for one platform serving both.

4.7. 7. Cross-repo sweeps + agentic readiness

- **Cross-repo sweeps** are the broad-simple corner of the maturity matrix — a single mandated change (CVE class, tracing rollout, auth wrapper migration) applied across many repositories. **Coordination cost, not per-repo change cost, dominates today.**
- Mechanism: one **sweep specification** → **fanout across the dependency graph** → one **ephemeral workspace per repo** → **batched PR waves** → one **aggregated evidence pack** → one human sign-off **at the sweep level**, not per PR.
- Safeguards: mandatory canary set (3–5% of scope), demonstrated sweep-level rollback, explicit failure budget with automatic pause and escalation.
- **Agentic readiness of the repository** — five tiers from Tier 0 (agent-hostile) to Tier 4 (agent-native with multi-repo change harness). Each tier unlocks the next class of automation. Investing in **the Tier 1 → Tier 2 lift** is often the highest-return platform work.
- **Open question.** Multi-service pre-deployment testing is unsolved industry-wide. The document names it as such rather than pretending otherwise.

Read the maturity-journey chapter when you are planning the platform investment or scoping a security-response automation programme.

4.8. 8. Measurement and evidence

- Every claim in the platform earns its keep on evidence — not on vendor promises.
- Categories the platform tracks:
  - **DORA basics** — throughput (deployment frequency, lead time) and stability (change-failure rate, MTTR).
  - **Human-attention-per-verified-outcome** by matrix cell — the meter the platform is measured on.
  - **Cognitive-burden proxies** — comprehension-record completion rate, reviewer self-report gap, mean time from first-look to sign-off.
  - **Agent-authored code share** — percentage of merged code authored by agents, and the **persistence rate** of AI-introduced defects (Liu et al. baseline).
  - **Agent-rewrite (slop) rate** — how often agent output is revised before merge and after merge.
  - **Task complexity to completion time** — by matrix cell, tracked quarter over quarter.
  - **Rework rate** — merged-then-reverted or merged-then-materially- modified within N days.
  - **Production issue rate** — attributable to changes at each autonomy tier.
  - **Auto-approval blast radius** — average files, services, and downstream callers touched by

an auto-merged change at Tier 0.

    - **Sweep throughput and coordination-hour reduction** — sweeps per quarter and human hours per sweep vs baseline.

- Consolidated in **Appendix T — Measurement Framework** with per- category definitions, cadences, leading-vs-lagging classification, counter-metrics, and a quarterly-review composition.
- Each Part carries a **How we measure this** subsection naming the 4-8 metrics most relevant to it, pointing back to Appendix T for the definitions. Appendix B holds the original evidence-and-metrics discussion.

Read Appendix T + the "How we measure this" subsections in every Part when you are turning the platform into an OKR conversation with the CFO or the head of engineering.

4.9. 9. Requirements are evidence too

- **The lifecycle does not start at the spec.** Everything downstream of the specification 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.
- **Cheap delivery does not improve the choosing.** Microsoft's experimentation platform reports about a third of tested ideas improve their target metric; Pendo puts ~80% of cloud features at rarely or never used. Triple the delivery throughput and you triple the bets placed at that same hit rate.
- **The material is already there and nobody reads it.** Tickets, sales calls, churn conversations, session replays, verbatims, funnel data. A worked example in the chapter: 900 tickets across three at-risk accounts become fourteen cited clusters in an hour — and the five customer calls that correct two of those clusters still take a product manager two days. The reading collapsed. The judgment did not.
- **Research recordings are the sharpest customer data in the building.** A customer agreed to a researcher listening, not to processing by a third-party model. What they agreed to travels with the recording and constrains retrieval. Deletion is the hard case: the recording goes, and the transcript, embedding, cluster, quoted brief, and slide all have to go with it.
- **The autonomy ladder applies upstream and stops one rung lower.** T0 reads and clusters, T1 drafts against citations, T2 proposes acceptance criteria, T3 files and deduplicates — T4 closed. A code change can be checked before it ships; a problem choice cannot, because the market answers months later. Live experiments with metric guardrails buy back a narrow class of change, named per class.
- **Intent debt** — the third debt. 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, so reviewers lost the prose signal they used to catch thin thinking. The record carries ticket IDs, sample size, and one of four states: hypothesis, evidenced, validated, invalidated.
- **Two objects extend the chain upstream** — signal and opportunity — and what production learns writes back to the opportunity that proposed it. An incident traces to the customer

evidence behind it; a specification with nothing linked can be stopped at the gate.

Read the requirements and discovery chapter when the platform is being justified on delivery throughput alone, or when someone asks where the specifications the agents work from actually come from.

4.10. 10. Your product has to be agent-ready too

- **The caller changed.** It arrives with a token, a task it did not write, no patience, and no eyes. It will not notice an odd result, will not stop at page one, and will combine your output with three other products' data before a human sees any of it.
- **Point authorization is not enough.** Point asks "may this principal read this resource." **Set authorization** asks which resources satisfy the query and the policy — and governs titles, snippets, citations, facets, counts, ordering, and whether a next page exists. A search endpoint is a security boundary.
- **Counts and facets are disclosure.** "Twenty investigations match this name" leaks with every record still hidden. Exactness is a claim about a named corpus and snapshot, not about the world.
- **Post-filtering is safe and still wrong.** Removing unauthorized items from a global top-k under-fills the authorized top-k, and no fixed overfetch factor fixes it. Revocation safety and grant freshness are separate guarantees that fail in opposite directions.
- **Derived artifacts keep their dependencies.** Embeddings, summaries, caches, and transcripts inherit source policy. A permissions-only change invalidates them even though the bytes are identical.
- **Delegation is an intersection, never a union.** Sponsor rights ∩ application capabilities ∩ agent charter ∩ purpose ∩ resource policy. Child agents narrow. A disabled tool has to reject direct invocation, because hiding it from discovery is not enforcement.
- **Cross-organization collaboration is ordinary, not an exception.** Each owner domain stays authoritative; the audience of a synthesis is the intersection of recipients permitted for every source in it.
- **What you owe the integrator**: a capability ledger separating announced from enabled from verified, a conformance suite they can run against their own tenant, three worked examples rather than thirty endpoints, tool schemas versioned like APIs, and errors designed for a machine that will retry.
- **Meter what agents consume.** Seat pricing assumes a human ceiling that agents remove. Attribute cost to the sponsoring principal, not to the integration's service account.

Read the product agent-readiness chapter when the company sells something other people's agents will call, or when a customer asks what your permission model does under agent load.

4.11. 11. Soft rails are the half nobody funds

- **Hard rails say no; soft rails say how.** The MCP gateway, autonomy tiers, constitutions, and the security control plane bound what an agent may do and are enforced. Skills, review commands,

hooks, scaffolds, and subagent definitions shape the default path through that space and are merely adopted. **An organisation that builds only hard rails gets compliance without competence** — every agent stays inside the fence and reinvents the same wheel differently.

- **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, and fails loudly when wrong. The measured 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 named (R387).
- **"Plugin" spans eight surfaces with different trust models.** A review command a person types and a hook that fires automatically with pipeline credentials are not the same risk object. Classify by who invokes it and whose credentials it runs with — not by packaging format — and set the review bar per class. IDE extensions sit outside the workspace broker entirely; CI steps run on pipeline credentials that are broad by default — OWASP ranks insufficient pipeline access controls fifth in its CI/CD top ten (R390).
- **A shared skill is executable context.** Every agent loads it, so a stale one produces last year's pattern confidently at volume, and a description saying "always run this first" is an instruction someone shipped. Pin versions, review descriptions as code, know your consumers, and be able to disable one everywhere inside an hour.
- **The registry is the product, not the catalogue.** Tiered publication (personal → team → provisional org-wide → org capability), expiry on the provisional tier, usage telemetry as the curation input, and discovery at the point of work.
- **Parity is a declared property.** The same capability on a laptop, in a workspace, in CI, and in an ambient agent — tested in each, not hoped for.
- **The incentive problem is real.** Nobody is promoted for writing a skill that saves another team a week. Usage telemetry turns "I wrote a thing" into "four hundred sessions used this", which a promotion committee can read.

Read the soft-rails chapter when agents are technically governed and still producing work that does not look like the organisation's.

4.12. 12. Not all correctness is automatically decidable

- **The evidence ladder is written in backend vocabulary**, and applied unchanged to every discipline it gives a false result — refusing to let UI work advance because the evidence does not reduce to a passing suite, or letting it advance on evidence that was never about the thing that matters.
- **UI is the clearest stress test, and it strengthens the thesis.** An agent produces real evidence for a UI change: visual diffs, interaction recordings, accessibility checks, design-system conformance, a browser and device matrix, a live preview. None of it encodes whether the interaction is confusing or whether the treatment communicates the product's intent.
- **The platform's job is to put a person in front of the running artifact with the automated evidence already gathered** — not to invent a score that makes the decision look decidable.
- **Autonomy ceilings are therefore per task class.** A dependency bump in a well-tested service reaches a high tier on evidence alone. A change to how a workflow feels does not, and saying so is more useful than pretending a threshold exists.

- **Cheap exploration pays unusually well here.** Three treatments generated against the same constraints, all three checked automatically, the human choosing an experience rather than implementing candidates one at a time.
- **The design system is a paved road.** Tokens, approved components, accessibility rules, reference implementations — organisational knowledge, and an agent left to infer UI primitives invents them the way it hand-rolls a retry loop when the auth wrapper is a wiki page.

Read the operating-models chapter §evidence is discipline-specific when someone argues that agentic development cannot work for design-led or research-led work.

5. Who reads what

Table 1. Navigation aid — which parts and appendices to read first, by situation

If you are…	Read these Parts	Read these Appendices
Deciding whether the platform investment is real	One-page argument + the principles chapter + the maturity-journey chapter (matrix + journey)	B (evidence and metrics), S (vendor positions)
Working out what this feels like day to day	The principles chapter on its own — ten invariants, the shortest chapter in the document	A (core design principles, what the platform must therefore provide)
Interviewing this platform against a vendor RFP	orchestrator + trust and governance + operating-models chapters + the maturity-journey chapter	I (PR review operating model), J (workspace reference), N (verification chain), S (vendor positions)
Adopting agentic development from scratch	orchestrator + maturity-journey + foundations + organizational + ambient-agents chapters	A (elaboration), G (presentation notes), L (MCP inventory), R (ambient catalog)
Auditing security posture	the trust and governance chapter (security, residency, IP, regulation, constitutions, customer data)	D (MCP focus notes), J (workspace reference), M (A2A component model)
Making the case to a CFO	One-page argument + the maturity-journey chapter + the trust and governance chapter cost/measurement subsection	B (evidence and supportable metrics), H (portfolio assessment)
Choosing where agent workspaces run	the operating-models chapter §placing agent workspaces + the maturity-journey chapter §agentic readiness	J (workspace reference), D (MCP focus notes)
Wiring MCP and A2A into an existing enterprise	the foundations chapter	D (MCP focus notes), L (MCP capability inventory), M (A2A component model)

If you are…	Read these Parts	Read these Appendices
Designing the org, career ladder, ambient portfolio	organizational + ambient-agents chapters	O (stakeholder decisions), R (ambient catalog), G (presentation notes)
Positioning the platform against Devin / Cursor / Copilot / Amp	the maturity-journey chapter + the trust and governance chapter	S (vendor positions)
Planning a cross-repo sweep programme	the maturity-journey chapter §cross-repo sweeps + the ambient-agents chapter	J (workspace reference), R (ambient catalog)
Landscape scan of the vendor / product space	the maturity-journey chapter	Q (landscape), S (vendor positions), U (commercial tools landscape), L (MCP inventory)
Security-reviewing or procuring a specific agent	the foundations chapter §properties of an agent for corporate use, §local and cloud MCPs	D (MCP focus notes), J (workspace reference), M (A2A component model)
Building the internal case with your own repositories	the maturity-journey chapter §agentic readiness	V (worth-surfacing repositories), H (portfolio assessment)
Building the shared capability layer	the soft-rails chapter + the foundations chapter	L (MCP inventory), T (measurement, Category J)
Exposing your own product to other people's agents	the product agent-readiness chapter + the organizational chapter	L (MCP inventory), D (MCP focus notes), M (A2A component model), T (measurement, Category I)
Running product discovery into an agentic delivery org	the requirements and discovery chapter + the maturity-journey chapter	K (spec-to-production object model), L (MCP inventory), R (ambient catalog, R.26)
Tracing any claim to its source	—	F (reference system), P (consolidated sources)
Presenting this internally in 60 minutes	One-page argument + the maturity-journey chapter + one anchor diagram per remaining Part	G (presentation notes), C (discussion agenda)

6. Where this differs from vendor pitches

Six major vendors (Cognition/Devin, Cursor, Augment, GitHub Copilot, Anthropic Claude Code, Sourcegraph Amp) are surveyed in **Appendix S — Vendor positions on the journey to agent autonomy**. Four consistent observations from that survey:

1. **Convergence on the what.** All vendors have adopted the framing that agents move from completion to orchestration. This debate is effectively over.

2. **Divergence on the how.** Vendors split on whether the organisational layer (governance, journey management, systemic measurement) is inside the product or the customer's job. Augment and Devin place more inside the product; Cursor, Copilot, Claude Code, and Amp leave it to the customer. **This document is one attempt to fill the gap that four of the six leave.**
3. **Systematic silence on debt and cognitive load.** No surveyed vendor engages the METR gap or the Liu persistence data publicly. Velocity metrics dominate. This document treats debt as first-class engineering concern.
4. **Systematic 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. This document treats customer data as an architectural concern, not merely a contractual one.

The 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**. This document is meant to be a credible plan for that organisational work.

7. Landmark numbers card

Six numbers to keep at fingertip level. Each is developed in Part Three, the maturity-journey chapter, or Appendix B and sourced in Appendix P. Two more belong on the card once discovery is in scope: **one-third** of tested ideas improve their target metric (Microsoft, R373) and **80%** of cloud-product features are rarely or never used (Pendo, R374).

Table 2. The six landmark numbers

Number	Claim	Source
20–40%	McKinsey 2020: technical debt is 20–40% of the value of a company's entire technology estate. The AI-native platform inherits this baseline.	R345
70–85%	Multiple studies of open-source and enterprise commit distributions place the routine slice (small corrective + simple adaptive + mechanical sweeps) in the 70–85% range. Covering this slice is where the largest lever is.	R346, R365 series
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.	R353
17% gap; 40% vs 65%	Anthropic RCT (52 engineers, per Osmani): AI-assisted participants scored 17% lower on comprehension (50% vs 67%); passive-delegation users scored below 40%, inquiry-driven users above 65%. Same tool, different discipline.	R350
+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.	R354

Number	Claim	Source
25% adoption → -1.5% / -7.2%	DORA earlier generative-AI analysis: 25% adoption increase associated with 1.5% decrease in throughput and 7.2% decrease in stability. A counterweight to unsupported velocity claims.	R3

8. What is still being argued about

A document this long can read as though everything in it is settled. Most of it is not. This section separates the claims that rest on judgement from the questions the field itself has not answered, so a reader knows which parts to push on.

8.1. Load-bearing claims this document cannot prove

Each of these is doing real work in the argument, and each rests on reasoning rather than measurement. They are the places to push first.

- **That attention is the binding constraint.** The premise everything else follows from. It is well-supported as a direction and has no clean measurement — nobody has published human-attention-per-verified- outcome for a real engineering organisation, which is precisely the meter this document proposes building.
- **That the 20–40% range is the honest upside.** Stated as a range rather than a number for a reason. Above it is marketing until measured; below it suggests the foundations are weaker than assumed. It is a calibrated judgement, not a finding.
- **That soft-rail coverage is uneven downstream of code generation.** An observation about the organisations the author has worked with, not a survey. The lifecycle-coverage metric rests on it.
- **That cross-repo sweeps are the under-served corner.** The economics are clear; the claim that the industry has neglected them is a read of the vendor market rather than a measurement of it.

8.2. Questions the field has not settled

These are live disagreements, not this document’s uncertainty.

- **Whether AI-assisted development is net faster at all.** The strongest skeptical result is still METR’s: experienced maintainers felt ~24% faster and measured ~19% slower (R354). Set against large-scale deployment reports claiming the opposite, the honest reading is that the disagreement is mostly about measurement methodology rather than about the data — and that a team optimising on developer self-report will over-invest either way.
- **Whether generated code degrades quality.** Contradictory results depend on context and evaluation criteria. Liu et al.'s 22.7% persistence figure (R353) is the strongest evidence that it does; it is one study, on public repositories, not a settled finding.
- **Where the boundary sits between an assistant, a tool-augmented model, and an agent.** Definitions differ across the literature, and autonomy scales proposed by different groups do

not align. This document's T0–T4 ladder is one such proposal and should be read as a position.

- **How agents get identity.** Treating an agent as an identity-bearing principal with its own access controls is argued here as a requirement. It is not yet common practice, and the gap between organisations running agents and organisations governing them as principals is wide.
- **What verification of generated work should look like.** The software engineering community names this as the central open problem, along with deciphering developer intent and the sheer volume of generated code entering codebases (R401). This document proposes the evidence chain as an answer; the field has not converged on one.
- **Whether multi-service pre-deployment testing is solvable.** Named as an open question in the maturity-journey chapter and unsolved industry-wide.

8.3. Where to find the rest

The epilogue lists what the document treats as hypothesis rather than argument. `OPEN-ISSUES.md` in the source repository records what is unresolved in the document **as a document** — unsourced claims, figures without editable sources, decisions the author has not made. The next section covers what is out of scope by choice, which is a different thing from unsettled.

9. What this document does NOT cover

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

- **Not a vendor evaluation.** Reference architectures + patterns, not a Copilot vs Cursor vs Codex vs Claude Code ranking. 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.** Cost, staffing, and rollout figures are ratios and evidence patterns, not a line-item budget.
- **Assumes baseline platform maturity.** Without paved roads, a service catalog, working CI/CD, and workload identity, the ambient-agents chapter reads as science-fiction; start with the foundations chapter.
- **Does not solve multi-service pre-deployment testing.** Named as an **open question** in the maturity-journey chapter. The industry has not solved it either.
- **Does not prescribe org design, model training methodology, or security certification mappings.** the organizational chapter names roles and permeability but not reporting lines; model choice is a procurement / routing / governance question; SOC 2 / ISO 27001 / FedRAMP mappings are the reader's job.
- **Data-science chapter is thin.** The platform primitives serve DS directly (pitch point 6) but a dedicated DS chapter would need its own doc. Named as such rather than pretended otherwise.
- **Does not prescribe a product-discovery methodology.** the requirements and discovery chapter 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. The provenance contract is the requirement; the method is the reader's.

10. Where to find things

The same argument is published at four depths. This companion is the shallowest; each step down adds mechanism, not new claims.

Table 3. The document set

Artifact	Length	What it is for
Full document	~390 pages	The audit trail — every mechanism, reference architecture, operating model, and consolidated source.
Concise edition	~245 pages	Every chapter of the full document with the operating-detail scaffolding subsections removed. Neutral variant only.
Abridged edition	~20 pages	The complete argument as one flowing long-form article. Same pitch points, same landmark numbers, no chapters.
This cliff-notes companion	~15 pages	The fifteen-minute skim, for deciding whether to read any of the above.
Hierarchical punchlist	33 pages	The work-breakdown: 23 workstreams, ~370 phase-tagged tasks.
Interactive citymap	browser	Pan-and-zoom map of ~350 components across the document structure, with cross-reference edges.

Full document (neutral): geekychris.github.io/ai-native-sdlc | **Case-study build:** /case-study.html | **Concise edition (~245 pages):** /concise.pdf | **Abridged edition (~20-page long-form article):** /abridged.pdf | **Punchlist (~300 tasks):** /punchlist.pdf | **Citymap (interactive):** /citymap.html | **Latest release (all artifacts):** /releases/latest | **Source repository:** github.com/geekychris/ai-native-sdlc