WHITE PAPER

FACTORY

Software Factory

An **Autonomy Maturity Model** for the enterprise: a systematic, measurable framework for bringing autonomy in continuous phases across the software development lifecycle, with human governance.

PREPARED FOR C-SUITE & ENGINEERING LEADERSHIP

FACTORY.AI

01 — Executive Summary

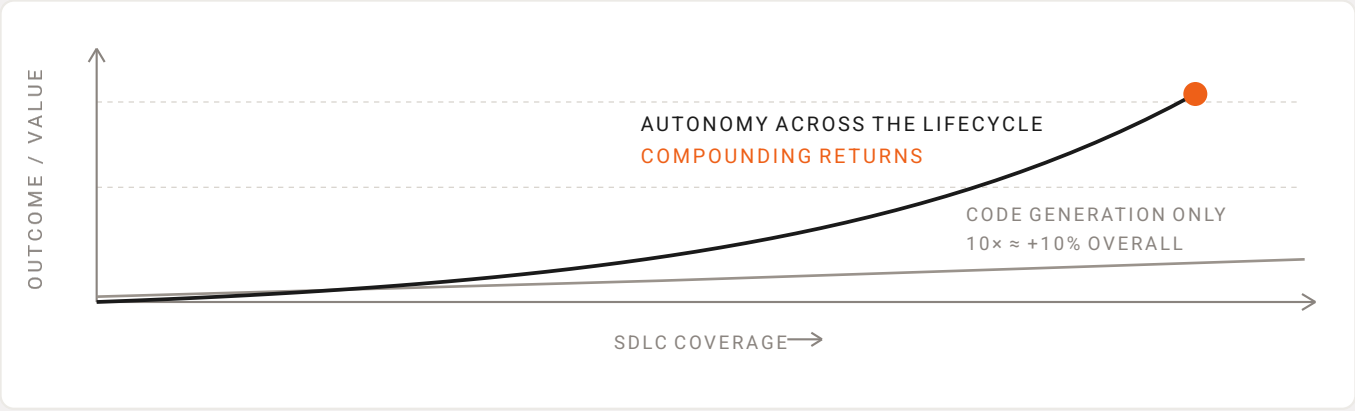
Software is increasingly produced by autonomous systems that maintain and improve themselves with minimal human intervention. The organizations that win the next decade will treat this as an operating-model transformation, not a tool purchase.

We call the end state a software factory: an autonomous system in which work flows from a signal or a goal (a ticket, a bug report, an alert, a KPI) through autonomous execution, review, and deployment, and back to completion tracking. Humans set intent and make judgment calls rather than performing mechanical implementation.

The Autonomy Maturity Model (AMM) measures how close an organization is to operating that factory. It is built on two dimensions (a technical foundation of Eight Pillars and a set of autonomous processes) scored across Five Maturity Levels, and improved through a repeatable methodology (DMASI) and a standardized deployment motion.

This matters because of where the value actually sits. Most AI investment today targets a single step of the lifecycle: writing code. A 10x improvement on that one step yields only roughly a 10% improvement overall, which is why many organizations are disappointed by the gains from code-completion tools. The return compounds only when autonomy moves the entire lifecycle (planning, review, testing, security, deployment, and operations), not one slot within it.

5	8	Level 3+
Maturity levels, from manual execution to a self-sustaining software factory	Technical pillars that make autonomous execution possible	The minimum bar for production-grade autonomous operation

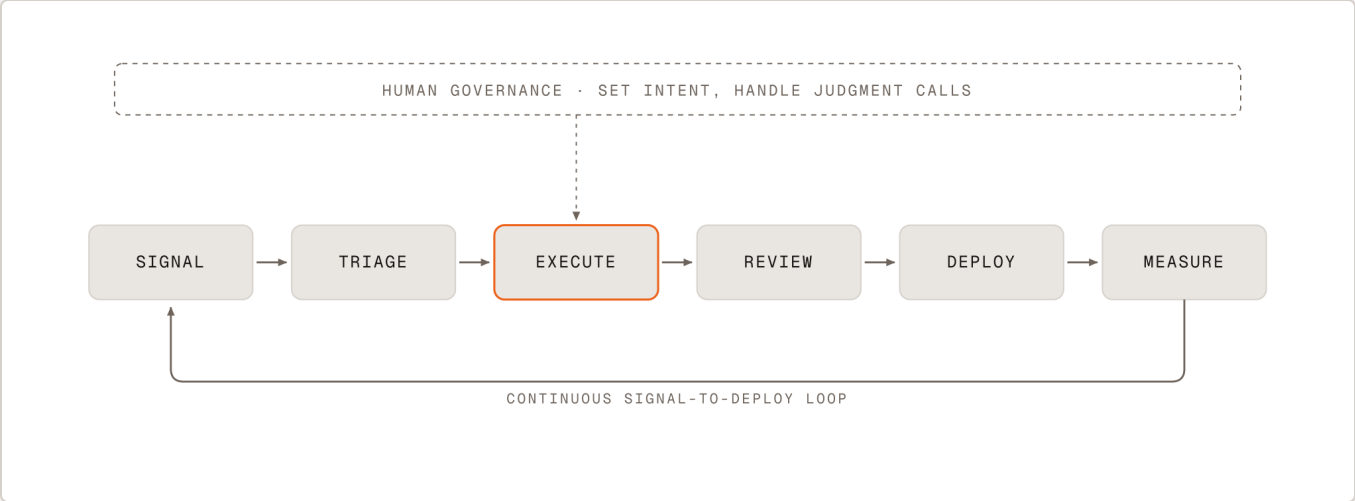


Returns stay flat when autonomy is confined to code generation, and compound only when it spans the lifecycle.

Why leadership should care. For the C-suite, the AMM converts a diffuse "AI strategy" into a concrete, board-legible metric: the percentage of active repositories that are agent-ready. For engineering leadership, it provides an objective, automatable scorecard that turns "adopt AI" into a prioritized backlog of infrastructure investments with measurable ROI.

02 — Software Factory

Autonomous software organizations build systems that maintain and improve themselves with minimal human intervention. Developers describe what they want built through whatever medium makes sense (a prompt, a ticket, a design file) and the system executes that vision with quality and precision: it generates idiomatic code, validates against linters, type checkers, and test suites, handles the pull request and review process, updates documentation, and deploys while monitoring for issues.



The continuous signal-to-deploy loop, with human governance setting intent across every stage.

The same pattern generalizes across the SDLC: a designer's mockup becomes an implemented, tested UI; a customer bug report becomes a diagnosed, fixed, and deployed change; a production alert becomes an investigated, mitigated, and documented incident; a tagged Jira/Linear story becomes a reviewed, merged pull request.

> **PRECISION, NOT PREVENTION**
>
> This framework enables precision in execution, not prevention of poor design. The system faithfully translates intent into working software, but it cannot save an organization from fundamental architectural mistakes or misaligned product decisions. It makes teams precise, not infallible, which is why human judgment becomes more valuable, not less.

When execution becomes automated, traditional role boundaries blur. The technical contributor shifts from mechanical implementation toward judgment and intent, deciding what to build and whether it is correct, while the factory handles the how. This is a transformation of the role, not its elimination: more people participate in building software, and each becomes dramatically more productive. The investment in skilled people pays off more, not less, when they operate with this kind of leverage.

03 — How the Model Works

How do you know if your organization is running a software factory? The AMM provides concrete, measurable criteria across two dimensions.

Dimension 1 · Technical foundation: The Eight Pillars

The infrastructure that enables autonomous operation: style and validation, build systems, testing, documentation, development environment and remote execution, code quality, observability, and security and governance. These provide fast feedback, clear instructions, reproducible environments, and observable behavior.

Dimension 2 · Autonomous processes

Human-agent collaboration and orchestration, triage, long-running goal agents, code review, incident response, documentation, security scanning, deployment, QA, model routing, and outcome measurement. Each maps to a concrete engineering workflow.

These processes are most powerful when they run on a shared foundation rather than as disconnected point solutions. When the same system performs code review, security analysis, documentation, and QA, context compounds: reviews are sharper because the system understands how the code was written, security analysis is informed by the review findings, and documentation reflects what actually shipped. This compounding effect, shared context across the lifecycle with centralized governance, is what a collection of separate tools cannot replicate.

Organizations progress through Five Maturity Levels, from manual execution to a self-sustaining software factory. The DMASI methodology (Define, Measure, Analyze, Scale, Integrate) plus a standardized deployment motion provide the systematic path to improvement.

04 — The Eight Pillars

PILLAR	WHAT IT PROVIDES & KEY SIGNALS
1. Style & Validation	Linters, type checkers, and formatters catch errors instantly. Signals: linter enforced, type checker active, formatter standardized, pre-commit hooks, validation under 30 seconds.
2. Build System	Deterministic build commands let agents verify changes before committing. Signals: documented and discoverable build, pinned dependencies, succeeds on fresh checkout, clear errors.
3. Testing	Fast unit and integration tests create tight feedback loops. Signals: core coverage, no external dependencies locally, unit tests under 2 minutes, clear diagnostics, flaky tests fixed.
4. Documentation & Instructions	Explicit instructions (e.g. AGENTS.md) capture tribal knowledge. Signals: discoverable setup/build/test/deploy commands, troubleshooting and architecture docs, updated with code.
5. Dev Environment & Remote Execution	Reproducible, sandboxed, increasingly remote and persistent environments. Signals: cold-start under 10 minutes, sessions survive disconnect, sandbox-by-default isolation for agents.
6. Code Quality	Modular code with clear boundaries helps agents reason within context windows. Signals: files under 500 lines, functions under 50, complexity under 15, clear boundaries, consistent naming.
7. Debugging & Observability	Structured logging, tracing, and metrics give runtime visibility. Signals: structured logging, distributed tracing, instrumented metrics, error messages with actionable context.
8. Security & Governance	Guardrails let agents move fast safely. Signals: branch protection, secret scanning, CODEOWNERS, dependency scanning, STRIDE threat modeling, sandbox enforcement, and sovereign deployment control.

SOVEREIGN DEPLOYMENT

Governance extends to the deployment model itself. Organizations should be the sovereign of their own deployment, choosing anywhere along a spectrum from fully managed SaaS, to regional deployments, to a self-hosted data plane, to fully air-gapped or on-premise, without handing over source code or ceding control. For regulated and security-sensitive enterprises, sovereign deployment is what makes autonomous engineering adoptable at all.

05 — Autonomous Processes

The Eight Pillars set the standard for how engineering organizations should operate so that autonomous processes can run on top. These workflows progress from manual execution to autonomous operation.

Human-Agent Collaboration & Orchestration

Developers work with agents to implement changes and, beyond 1:1 interaction, orchestrate multiple agents in parallel. Orchestration deepens along a spectrum: reliable single-task execution, intra-task parallelization, inter-task parallelization (including tickets auto-picked up from Linear/Jira), and finally automatic decomposition of complex requirements into a directed acyclic graph (DAG) of subtasks. Long-running, multi-day autonomous execution sits at the frontier.

Triage & Signal-to-Action

Incoming signals across Slack, GitHub, Linear, and support are triaged into well-formed, deduplicated, correctly-routed work items: the entry point to the factory, with time-to-triage under 24 hours.

Long-Running Goal-Oriented Agents

The frontier: an agent given a goal that does not end (own a surface or hold a KPI) that triages, builds, monitors, and notifies continuously, escalating only judgment calls.

Code Review, Incident Response, Documentation & Security

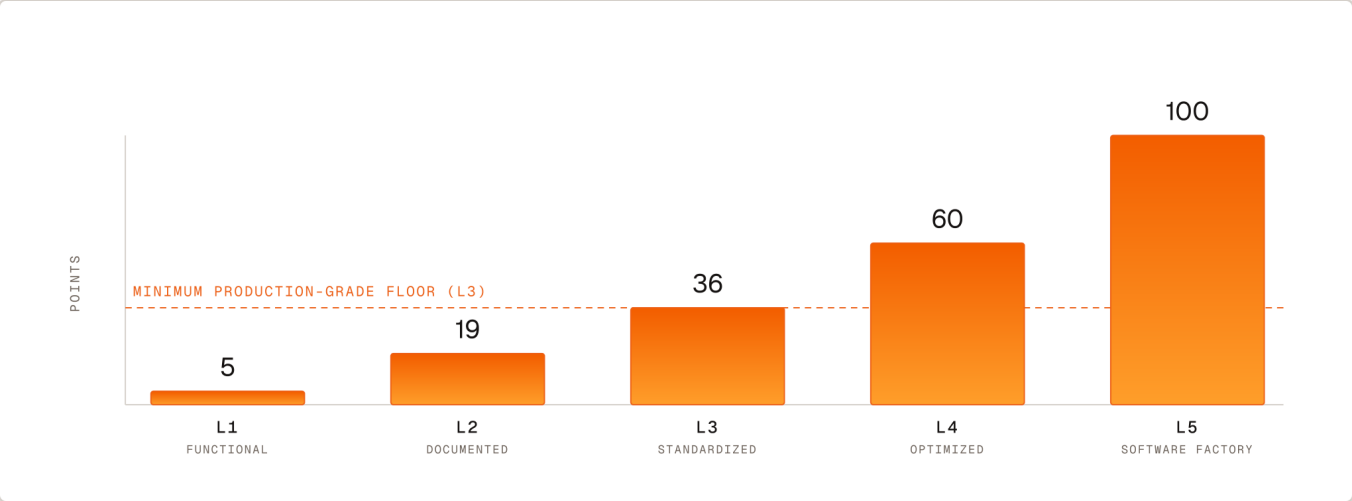
Autonomous review provides feedback within minutes and lets humans focus on architecture and business logic. Incident response detects anomalies, correlates errors, and remediates or prepares context. Documentation stays current because it is generated within the same change as the code. Security scanning runs continuously on every commit. Representative targets: review comments within one minute, mean time to detection under two minutes, documentation freshness above 90%, and zero secrets committed.

Deployment, QA, Model Routing & Outcome Measurement

Deployment becomes progressive and low-touch (staged rollouts with automatic rollback). QA shifts to dynamic, change-aware validation that drives the product like a real user. Model routing selects the optimal model per task with multi-provider fallback, so a single provider outage does not stop work and the organization is never locked to one model lab; because different models lead on different tasks, automatic routing places each task on the Pareto-efficient frontier of quality and cost. Outcome measurement tracks business results (cycle time, autonomy ratio, incident MTTR, cost per change) rather than raw activity.

06 — The Five Maturity Levels

Organizations do not become autonomous overnight. They progress through five distinct levels, each a qualitative shift in how work gets done. Point values scale exponentially because higher-level capabilities require significantly more investment.



Scores are non-linear: each level demands disproportionately more rigor than the last.

5 POINTS

Level 1: Functional L1

Code runs, but requires manual setup and lacks automated validation. Signals: README present, linter configured, type checker active, formatter standardized, unit tests exist.

19 POINTS

Level 2: Documented L2

Workflows are written down and some automation is in place. Signals: an AGENTS.md, reproducible dev environment, pre-commit hooks, documented build, branch protection, structured logging, CODEOWNERS.

36 POINTS · MINIMUM PRODUCTION-GRADE BAR

Level 3: Standardized L3

Processes are defined, documented, and enforced through automation. Signals: end-to-end tests, up-to-date documentation, active security scanning, distributed tracing, and metrics collection.

60 POINTS

Level 4: Optimized L4

Fast feedback loops and data-driven improvement. Sub-minute feedback, comprehensive observability, inter-task parallelization with tickets auto-picked from Linear/Jira, and automated review, QA, and triage contributing meaningful capacity. Remote execution is the standard.

100 POINTS · THE FRONTIER

Level 5: Software Factory (autonomous) L5

A self-improving system with sophisticated orchestration. Complex requirements decompose into DAGs and execute in parallel, multi-day missions run reliably, and squads of goal-oriented agents hold standing goals. People manage portfolios of work and set intent. Few organizations reach this level.

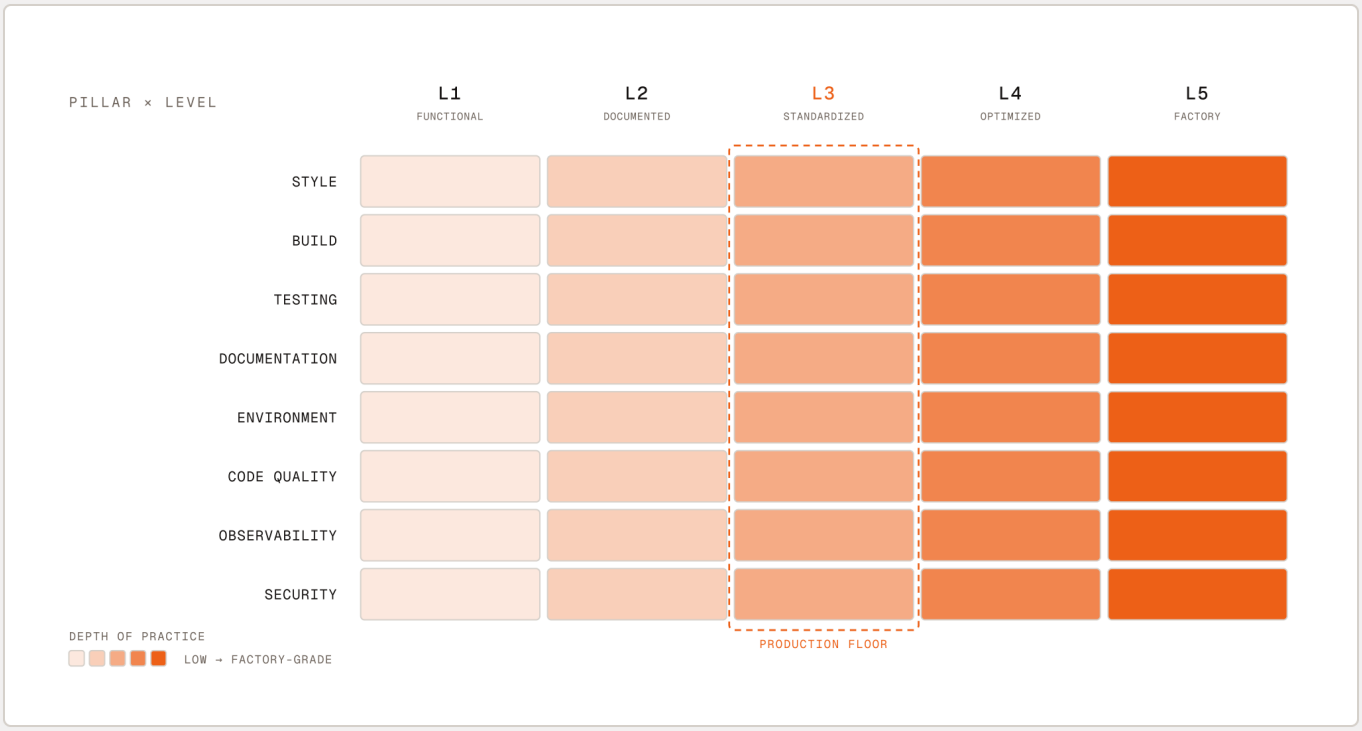
A CANDID NOTE ON THE FRONTIER

Even the most advanced organizations are only a small fraction of the way to fully autonomous signal-to-deploy. This is a multi-year journey, and progress, raising the floor level by level, matters more than reaching the destination. The model is designed for continuous phases under human governance, not a single leap to full autonomy.

07 — Measurement & Scoring

Repositories progress through the levels by accumulating points from completed signals. Each signal carries a point value based on difficulty and impact: Level 1 signals are worth 1 point each, Level 2 worth 2, Level 3 worth 4, Level 4 worth 8, and Level 5 worth 16.

LEVEL	THRESHOLD	WHAT IT REPRESENTS
Level 1	5 points	Basic tooling every repository should have
Level 2	19 points	Documentation and reproducible builds
Level 3	36 points	Standardized, production-grade autonomous operation
Level 4	60 points	Measured, optimized, parallelized execution
Level 5	100 points	Self-improving, fully orchestrated software factory

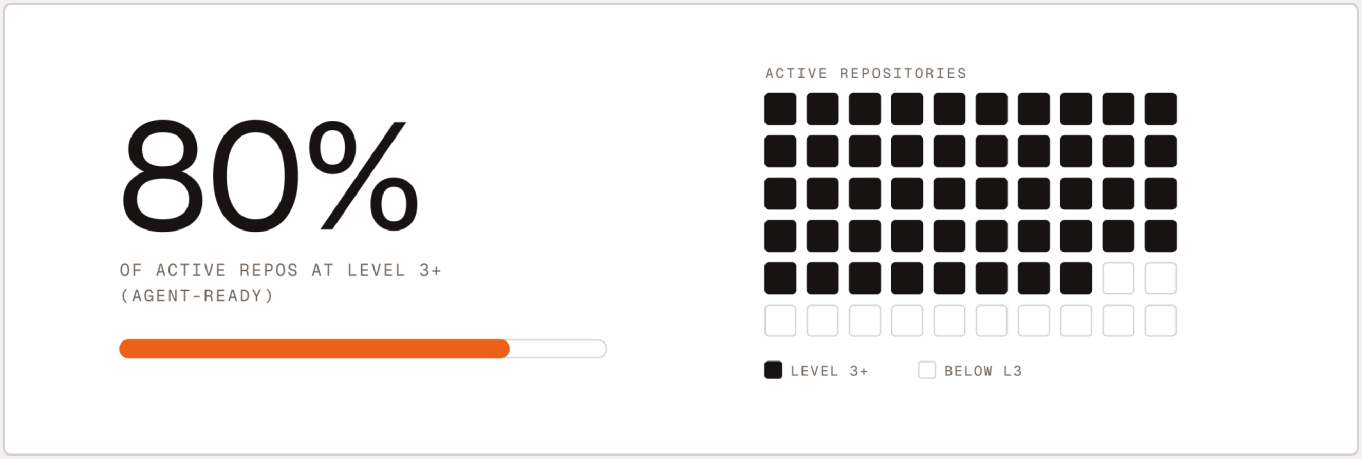


Depth of practice deepens left to right. The Level 3 column is the production-grade floor every pillar must clear.

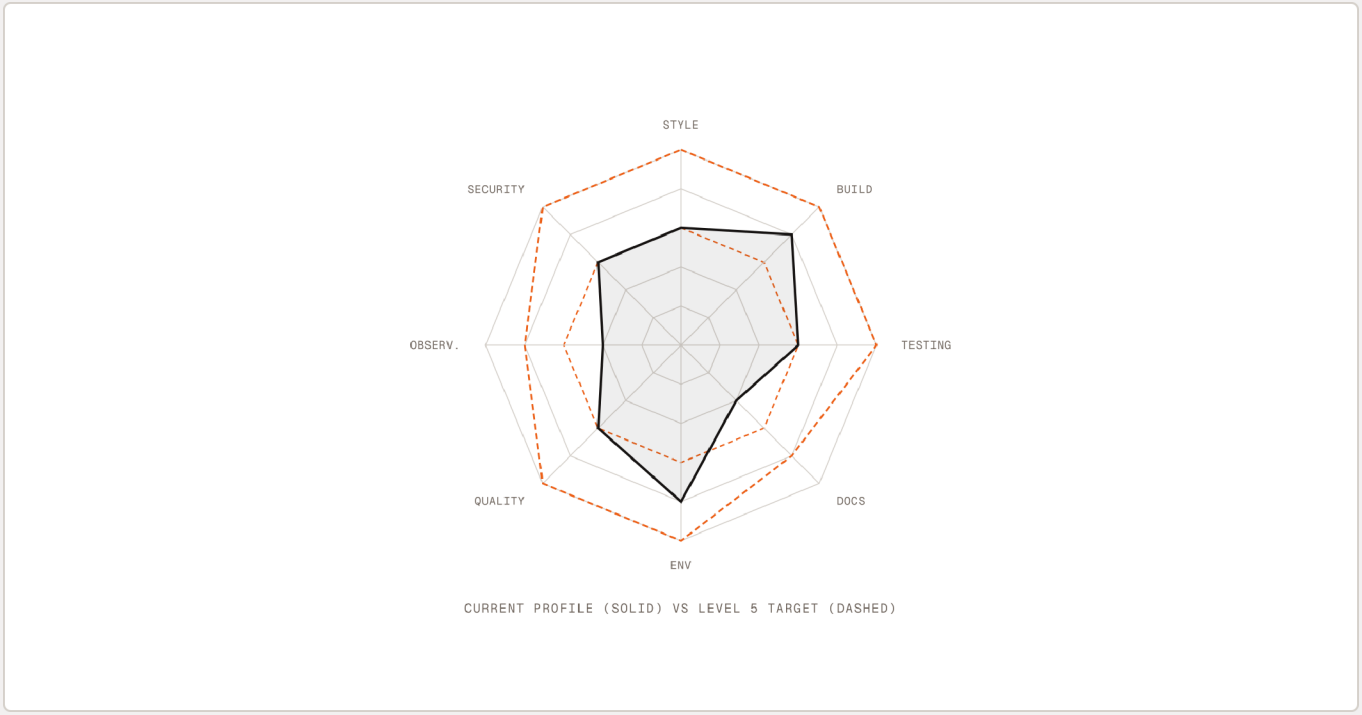
Signals are deliberately binary: present or absent, no partial credit and no subjective judgment. They should be automatable, actionable, relevant to autonomous execution, and fast to measure (under five minutes per repository).

THE HEADLINE METRIC FOR LEADERSHIP

Organizational Readiness = (Repositories at Level 3+ ÷ Active Repositories) × 100. “80% of our active repos are agent-ready” is clearer and more actionable than “our average score is 73.2%.” The goal is to raise the floor, not to achieve perfect scores everywhere.

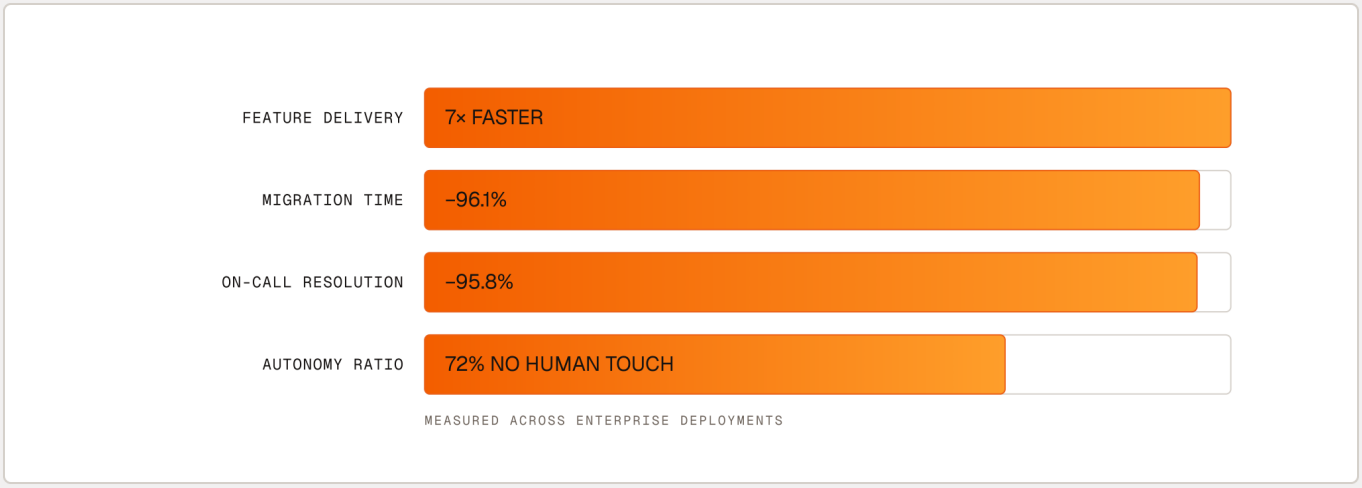


Portfolio view: the share of active repositories at Level 3 or above, ready for autonomous work today.



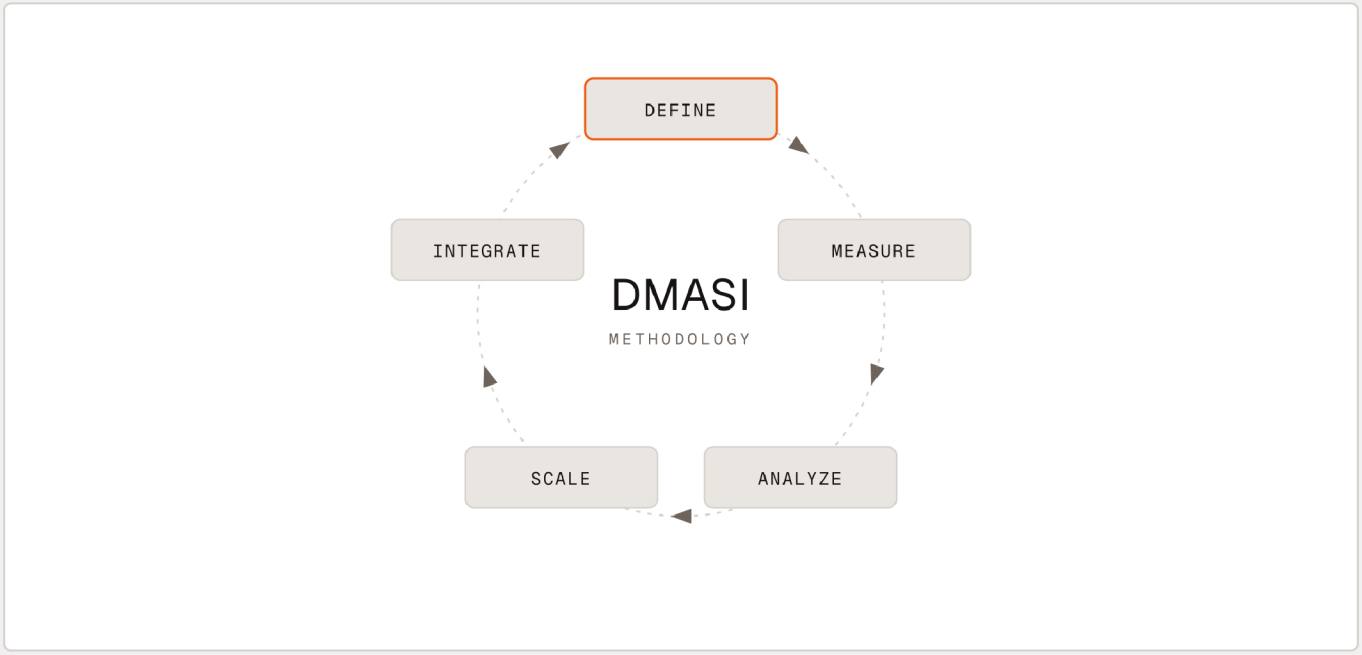
Per-repository capability profile: current versus the Level 5 target across all eight pillars.

Readiness scoring is a leading indicator; it measures whether the foundation is in place. It should be paired with lagging, business-level outcome metrics that prove the foundation is paying off: cycle time, autonomy ratio (the share of work completed with minimal human intervention), code shelf life, incident MTTR, and cost per change. The objective is to maximize outcomes, not activity.



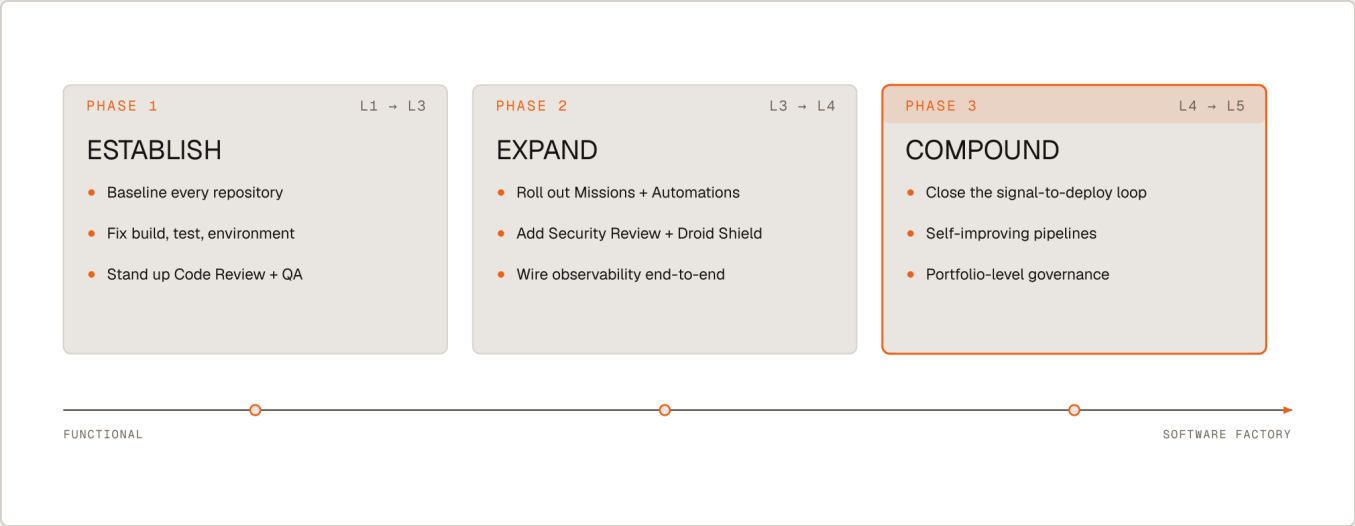
Outcomes from enterprise deployments. Spend maps to merged work, not tokens consumed.

08 — DMASI Methodology & Deployment Motion



DMASI turns readiness into a continuous improvement cycle, not a one-time audit.

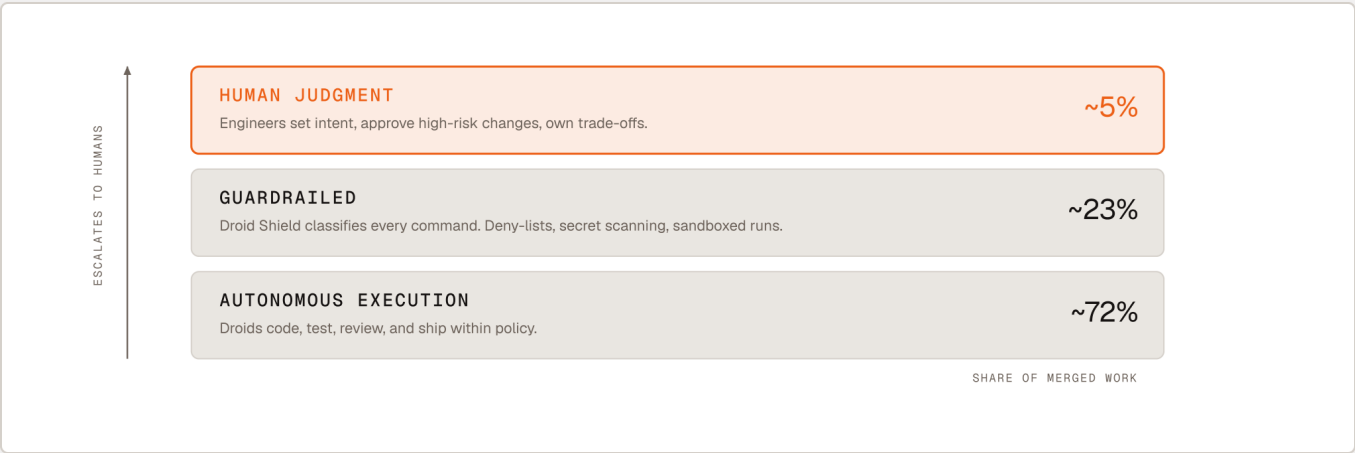
PHASE	OBJECTIVE
Define	Establish which repositories, teams, or systems to measure and set target maturity. Start with high-value, actively-developed repositories.
Measure	Implement automated, non-disruptive measurement of the Eight Pillars and collect baseline data across the defined scope.
Analyze	Identify the biggest gaps, common missing elements, and the improvements that would unlock the most value.
Scale	Apply improvements systematically with templates, automated remediation, and agents, so new repositories start at Level 3, not Level 1.
Integrate	Embed continuous measurement and improvement into standard workflows so the system maintains its own health.



A phased rollout: reach the production floor, scale autonomy, then close the loop.

Deployment Motion

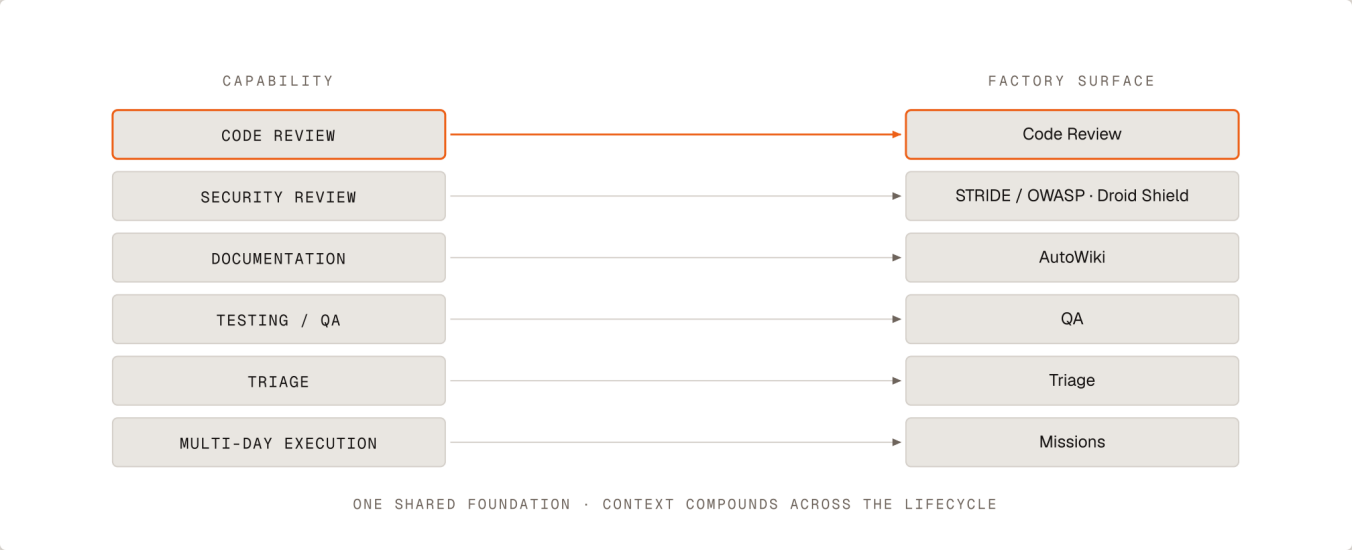
Every deployment is structured around a fixed set of software factory capabilities rather than ad hoc requirements. A customer's user journeys are mapped to a shared capability checklist; both sides sign off on the same document; success is measured against it. Prove one team's full software factory deployment end-to-end, then replicate across every other team, with a dedicated owner accountable for the multi-month lifecycle from one validated team to the whole organization.



Autonomy expands within guardrails. Judgment calls escalate to engineers; everything else runs autonomously.

09 — Capability Map

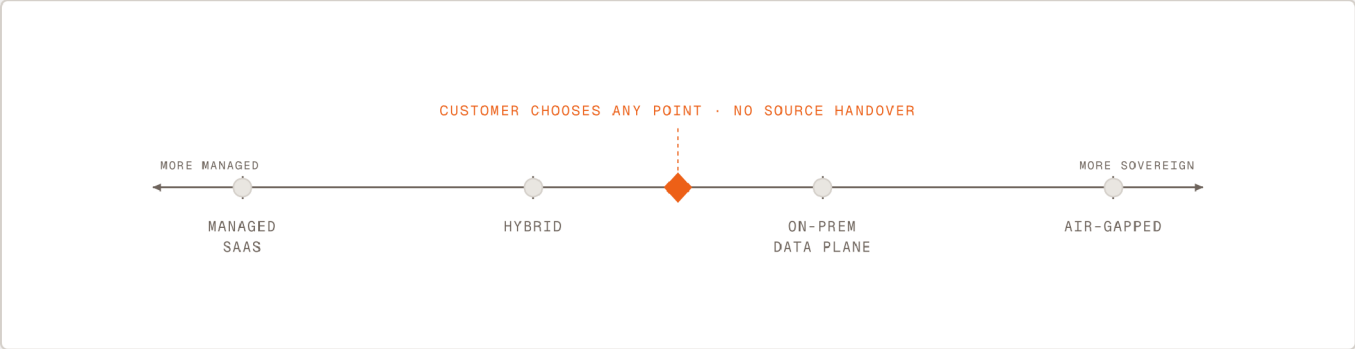
Each capability in the model maps to a concrete engineering surface and a corresponding Factory product.



Capabilities and the Factory surfaces that deliver them, on one shared foundation.

CAPABILITY	FACTORY PRODUCTS	WHAT IT DELIVERS
Synchronous agents	Droid CLI and Desktop	Interactive, in-editor and CLI execution of developer intent
Remote / persistent execution	Droid Computers	Always-on environments that survive disconnects and run long work
Multi-agent automations	Squads	Coordinated squads of agents working a shared objective
Multi-day autonomous execution	Missions	Orchestrator/worker missions spanning hours to days
Task decomposition / DAG	Missions and Automations	Automatic decomposition of complex work into scheduled subtasks
Triage / signal-to-action	Triage	Well-formed, deduplicated, correctly-routed work items
Long-running goal agents	Triage	Standing goals executed continuously against a KPI
Code review	Code Review	Automated, exhaustive review feedback on every change
Incident response	/incident and QA	Detection, correlation, and remediation with documentation
Documentation	AutoWiki (/wiki)	Continuously generated, always-current documentation

CAPABILITY	FACTORY PRODUCTS	WHAT IT DELIVERS
Security review	Security Review (STRIDE and DroidShield)	STRIDE-informed analysis and continuous scanning
Testing / QA	QA (/install-qa, agent browser, and computer QA)	Dynamic, change-aware QA that drives the product like a user
Deployment / release	Release automation	Progressive rollouts with automatic rollback
Model routing	Router	Per-task selection on the Pareto frontier of quality and cost
Outcome measurement	Analytics (/agent-effectiveness-report)	Business-outcome reporting per team and surface
Agent readiness scoring	Agent Readiness	Org-shareable readiness reports with prioritized remediation
Sovereign deployment	SaaS, regional, self-hosted data plane, air-gapped, and on-prem	The customer controls the deployment
Enterprise governance	CMEK, audit logging, EU single-region, sub-orgs, and service accounts	Encryption, audit logging, regional isolation, and model-access management



From managed SaaS to air-gapped on-prem. AWS, Azure, GCP. SOC 2, GDPR, ISO 42001.

10 — Getting Started

The path to a software factory is incremental and measurable:

1. Baseline. Measure the Eight Pillars across active repositories and compute organizational readiness (the share at Level 3+).
2. Prioritize. Target the highest-value repositories and the pillars with the most common gaps.
3. Remediate at scale. Use templates, defaults, and agents to raise the floor, so new repositories start at Level 3.
4. Prove one team end-to-end, then replicate the deployment motion across the organization.
5. Integrate measurement into standard workflows so readiness improves continuously without manual intervention.

> THE OBJECTIVE
>
> The goal is not to reach Level 5 everywhere. It is to raise the organizational floor to Level 3+ predictably, and to let autonomous processes compound from there, under human governance, in continuous phases.

GET STARTED

Measure your Agent Readiness, then close the gap to Level 3 and beyond.

SOC 2 GDPR ISO 42001 CCPA DROID SHIELD

FACTORY.AI ↗ · CONTACT SALES