Genius Software Manager

An Evidence-Based Architecture for Measuring Software Engineering Productivity, Quality, and Business Outcomes in AI-Assisted Development

Kenneth Hurley
Genius Ventures, Inc.
Technical Architecture and Research Framework

October 27, 2026 • *Design paper; no empirical results claimed*

Abstract

Artificial intelligence has reduced the effort needed to generate code, tests, and documentation, exposing the limits of developer productivity measures based on commits, lines of code, or hours worked. We present Genius Software Manager (GSM), a proposed evidence-based architecture that measures software engineering outcomes independently of the tools used to produce them. GSM correlates version-control, verification, review, deployment, and incident records into auditable workstream and service histories. Its measurement model separates (i) five DORA software delivery measures at the team/service level, (ii) complexity-adjusted verified outcomes, quality, velocity, and durability at the workstream level, and (iii) business impact and cost. The architecture uses C++ domain plugins, OpenAPI contracts, RocksDB, locally controlled evidence, and cryptographically signed report manifests. GSM neither hosts nor runs coding agents, GSD, or other development tools; such tools remain external and optional sources of observable evidence. We describe the proposed metrics, trust boundaries, limitations, and a prospective evaluation protocol. The paper reports an architecture rather than observed productivity gains.

Index Terms—software engineering productivity, DORA, AI-assisted development, human supervision, evidence provenance, software delivery, engineering economics.

1 Introduction

Software engineering productivity is not equivalent to source-code production. An engineer may eliminate a major defect with a small change, while thousands of new lines may increase maintenance cost. AI-assisted coding makes the mismatch stronger: generation throughput can rise without an equivalent increase in correct, useful, durable outcomes. The SPACE framework argues against reducing developer productivity to any one activity metric [1]; developer-experience research likewise points to the role of friction, feedback, and cognitive load [2].

GSM is designed to answer a narrower, testable question: *How much verified, durable engineering value is delivered, how effectively does a team deliver it, and how does AI assistance affect those outcomes?* It treats source events as evidence, not as intrinsic value. The approach is intended to support engineering decisions, transparent client reporting, and research into AI-assisted workflows, rather than employee surveillance.

The paper makes four contributions: (1) a tool-independent C++ evidence architecture; (2) separate, inspectable measurement models for delivery health, workstream outcomes, and business value; (3) auditable provenance and signed reports; and (4) an evaluation protocol designed to distinguish correlation from causation. These are design contributions, not proven experimental findings.

2 Scope and Architectural Decisions

GSM is a **software delivery intelligence system**, not an integrated development environment, agent supervisor, coding-agent runner, or replacement for GSD. External tools may include GitHub, CI systems, deployment tooling, monitoring, GSD-produced files committed to a repository, and other development systems. GSM reads their available records through adapters; it does not start GSD sessions, spawn Claude Code or Codex, route agent prompts, or depend on TypeScript/Bun execution.

The canonical backend uses C++17, OpenAPI-defined contracts, domain plugins, and RocksDB. A lightweight static browser application talks to the C++ REST service. A Linux host maintains local evidence files, storage, report bundles, and signing keys. Remote access through Cloudflare is optional; neither public cloud storage nor a hosted database is a prerequisite.

3 Evidence Acquisition and Provenance

3.1 Normalized evidence model

Each collected record should contain a stable event identifier, event type, source and source identifier, occurrence and ingestion times, schema version, payload digest, and relevant entity links. Such links may include organization, team, repository, service, workstream, commit, pull request, deployment, and incident identifiers. Source references must remain resolvable or have a documented retention policy.

Source adapters validate inbound webhooks or vendor responses, deduplicate retries, classify environments, and nor-

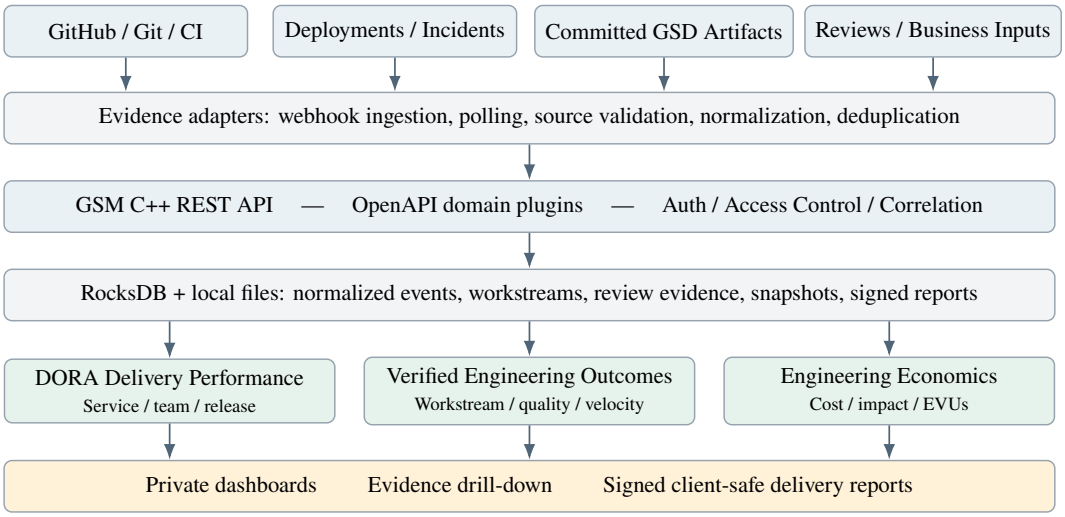


Fig. 1: Proposed logical architecture. All development and agent tools remain outside GSM. Adapters ingest observable source records; a C++ backend normalizes evidence for separate measurement and reporting domains. Solid boxes are architectural components, not a claim that every service is implemented.

malize events before calculation. Measurement engines operate on retained GSM evidence rather than consulting vendors at calculation time. This permits replay under a fixed metric version and audit of the precise event population that produced a result.

3.2 Correlations and uncertain evidence

Correlation is an explicit data object, not a guess from chronological proximity. Deployment-to-change mapping should prefer artifact manifests, build provenance, and exact commit sets. Incident-to-deployment mapping should prefer explicit links and rollback records. Time-window matches may be suggestions but require approval when causal evidence is uncertain. Each correlation retains its method, approval, and confidence.

3.3 Integrity and local control

The backend's nine initial storage domains cover workspaces, repositories, jobs, events, agent sessions, reviews, proof snapshots, published reports, and settings [5]. Additional domains are proposed for deployments, services, incidents, metric snapshots, workstreams, and economic assessments. Append-only audit behavior must be enforced above RocksDB's mutable key-value interface. A content digest can reveal changes to a captured record; it cannot establish that the original external record was accurate.

4 Three Measurement Domains

GSM separates three questions that share evidence but differ in unit of analysis (Fig. 3). **Delivery performance** describes a team's service releases and recovery. **Engineering outcomes** describe accepted and verified workstreams. **Engineering economics** estimates costs and benefits associated with those

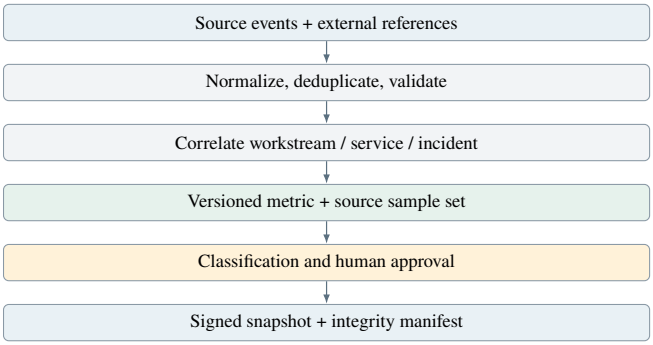


Fig. 2: Proposed evidence lineage. Metric results remain traceable to normalized records, source references, classification decisions, and formula versions.

outcomes. A high release count is not proof of high engineering value; a high-value architectural change may involve relatively few releases.

5 Team and Service Delivery Performance

GSM proposes the current five-measure DORA software delivery performance model [3, 4]. The scope is an organization, owning team, service, production boundary, and time window. A service may span multiple repositories, and a repository may contain multiple services. Eligible production delivery events must therefore be identified by service and actual release or user-exposure boundary, rather than branch or repository alone.

For eligible delivered change i, lead time is $L_i = t_i^{\text{delivery}} - t_i^{\text{commit}}$; medians, higher percentiles, and sample counts are retained. Deployment frequency is $N_{\text{success}}/\Delta t$ or its equivalent interval. Recovery duration for deployment-caused impair-

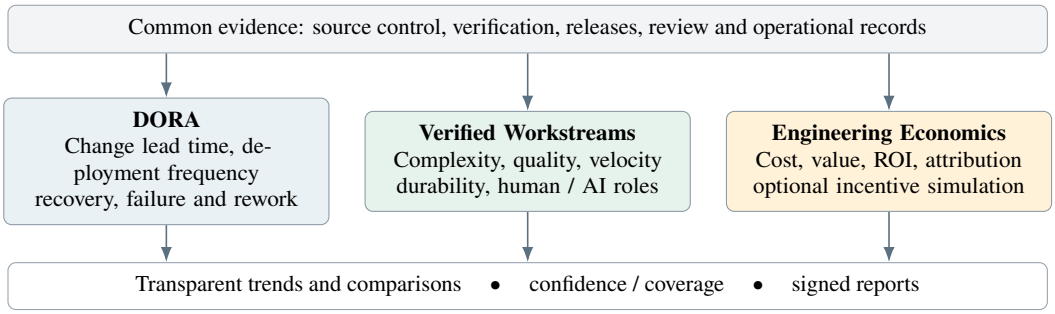


Fig. 3: The three measurement domains and their distinct units. DORA is not an individual score or a direct compensation input. Shared evidence permits drill-down without collapsing the measures into one rank.

Table 1: DORA measures tracked separately

Measure	Operational meaning
Change lead time	Commit/change set to successful production delivery
Deployment frequency	Successful releases per window or time between releases
Failed deployment recovery time	Impairment start to service restoration
Change fail rate	Failed-change deployments / eligible deployments
Deployment rework rate	Unplanned remediation deployments / eligible deployments

ment j is $R_j = t_j^{\text{restored}} - t_j^{\text{impaired}}$. The two instability ratios are

$$\text{CFR} = \frac{N_{\text{failed change}}}{N_{\text{eligible deployment}}}, \tag{1}$$

$$\text{DRR} = \frac{N_{\text{remediation deployment}}}{N_{\text{eligible deployment}}}. \tag{2}$$

A deployment causing an incident and the subsequent repair release are distinct events and may contribute to different numerators. Denominators, exclusions, sample sizes, and metric-profile versions remain visible. Zero denominators produce "not available," not a spurious zero rate. Unresolved incidents are not recorded as instant recoveries.

DORA metrics describe *team/service delivery systems*. A failed deployment must not be assigned to an individual based on commit ownership; the workstream quality model can consider a linked defect only after credible correlation and review [8].

6 Verified Workstreams and Complexity

The primary unit of outcome-level engineering measurement is a **verified workstream**: a coherent feature, migration, GSD phase, defect correction, security remediation, or architecture change with accepted scope and evidence of completion. Required evidence may include specifications, linked changes, tests, reviews, approvals, release records, and later operational observations. Agent execution or PR creation alone does not establish verification.

The proposed workstream complexity rubric has five levels: 1 (narrow, low risk), 2 (bounded feature), 3 (cross-component change), 4 (subsystem or substantial operational workflow), and 5 (architecture-level or critical system work). A planning-time proposal and verification-time human approval reduce post hoc inflation. Similar workstreams may be grouped to prevent artificial splitting.

For verified workstreams i within a period Δt, one base throughput quantity is

$$V_{\text{base}} = \frac{\sum_{i=1}^{n} c_i \, I_i}{\Delta t}, \quad I_i \in \{0, 1\}, \tag{3}$$

where c_i is the approved complexity weight and $I_i = 1$ only when the verification threshold is met. This is not a universal exchange rate between unlike engineering tasks; the reported comparison cohort must preserve work type, risk, and role context.

7 Quality, Velocity, and Durability

7.1 Quality

The initial proposed quality score $Q \in [0, 100]$ combines six higher-is-better component scores:

$$\begin{aligned} Q = {} & 0.25D + 0.20R + 0.20T \\ & + 0.15F + 0.10G + 0.10A, \end{aligned} \tag{4}$$

where D describes escaped-defect performance, R avoidable rework, T relevant verification evidence, F review findings and severity, G regression/rollback performance, and A architectural conformance [6]. The component rubrics and weights are policy choices requiring empirical calibration. Tests and review comments are evidence of quality, not quality by count. Pre-release findings must be distinguished from escaped production defects.

7.2 Velocity

The proposed normalized velocity score uses complexity-weighted verified outcomes (40%), cycle time (20%), planned

scope completed (15%), sustained cadence (15%), and cross-system delivery breadth (10%) [6]. This score remains separate from DORA deployment frequency. Stage-time decomposition can distinguish implementation time from waiting on reviews, builds, and releases.

7.3 Durability

The model proposes 30-, 90-, and 180-day follow-up windows. Relevant signals include attributed incidents, rollbacks, repeated corrections, and rewrites caused by implementation failure. A planned redesign or intentional simplification is not automatically a defect. Absence of incidents is weak evidence when software has had little operational exposure; durability reports should include exposure and evidence coverage.

7.4 Composite effectiveness

An optional summary combines quality, velocity, collaboration C, and planning accuracy P:

$$E = 0.50Q + 0.35V + 0.10C + 0.05P. \tag{5}$$

The interface must always reveal Q and V separately; a blended score must not conceal high failure rates behind speed. This score is a proposal for study, not a validated individual ranking instrument.

8 Human Contribution and AI Leverage

Workstream contribution is allocated by engineering role, not solely by commit author. Roles include architecture, specification, implementation, AI-output supervision, review, verification, and delivery acceptance. An approved attribution ledger records contributors, shares, evidence links, reviewers, and revisions; for a workstream, shares sum to unity. AI tools are not pay recipients.

GSM need not inspect agent prompts or host agent sessions to recognize AI assistance. AI metadata may be recorded where supplied by trustworthy external sources; otherwise the level of AI involvement is "unknown," not automatically "human-only." This distinction avoids turning data absence into a favorable comparison.

One proposed AI Leverage Factor is

$$\mathrm{ALF} = \frac{V_{\text{quality adjusted,AI}}}{V_{\text{quality adjusted,baseline}}}. \tag{6}$$

The denominator may come from matched earlier work or comparable teams. This is a descriptive ratio; uncontrolled comparisons do not prove that AI caused an improvement. Comparisons must account for task mix, service maturity, contributor roles, test coverage, and review burden.

9 Business Value and Engineering Economics

The proposed economic domain relates verified outcomes to fully loaded engineering costs and business impact [7]. Full cost can include compensation, tools, infrastructure, benefits, and allocated overhead. Examples of useful organizational aggregates are cost per verified workstream, cost per verified complexity point, gross margin per employee, and engineering investment relative to revenue. The denominators and allocation methods must be disclosed.

Engineering Value Units (EVUs) provide a *policy-defined relative measure*, not independently realized monetary value:

$$\mathrm{EVU}_{\text{base}} = S_{\text{verified}}\, C_{\text{weight}}, \tag{7}$$

$$\mathrm{EVU}_{\text{adjusted}} = \mathrm{EVU}_{\text{base}} M_Q M_D M_B M_T, \tag{8}$$

with quality, durability, impact, and timing multipliers. Inputs and formula versions are retained; business-impact levels require review. Estimated and realized financial values must not be mixed without labels. Where defensible realized value B and allocated cost K exist, ROI $= (B - K)/K$; otherwise the result is an estimate or unavailable.

Outcome-based compensation is a *separate policy layer*. Shadow calculations may model milestone payments, bonuses, or incentive pools, subject to human approval, appeals, confidentiality, and labor rules. Service-level DORA scores should never flow directly into individual pay calculations. An unvalidated EVU model must not be treated as a payroll oracle.

10 Signed Evidence Reports and Trust Boundaries

A delivery report can present the requested scope, accepted result, verification checks, reviewer decisions, source references, and approved metric summaries. GSM proposes canonical serialization, SHA-256 digests of report-eligible events and snapshots, and Ed25519 signatures over a report manifest [5]. The public view exposes only sanitized records; raw logs, private repository content, incident details, signing keys, and compensation remain local or access-controlled.

A valid signature establishes integrity and the claimed signing identity, *not* correctness, causal attribution, or economic value. Source accuracy still requires substantive checks. Reports must preserve metric version, sample coverage, approvals, and exclusions, allowing reproducible calculations. Protected admin endpoints remain distinct from limited public verification endpoints.

11 Research and Evaluation Plan

The research program should proceed in stages rather than reporting gains before data exist. First, compare ingested and normalized events against independently sampled Git/CI/incident

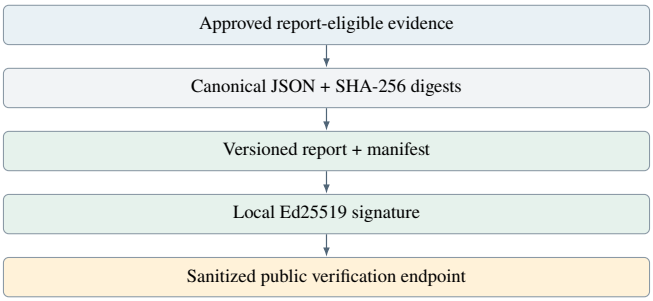


Fig. 4: Report integrity process and distinct access boundaries. Signing detects changes after signing but does not prove that external evidence was truthful.

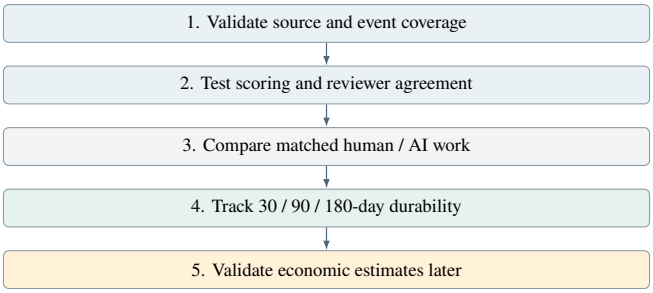


Fig. 5: Proposed validation sequence. Each stage tests the reliability of inputs and interpretation before higher-stakes use.

records. Report source completeness, duplicate rate, mapping precision, and disagreement over failure or rework classification. Second, have independent reviewers score workstream scope, complexity, and quality; measure inter-rater agreement and resolve rubric ambiguity.

Third, use matched cohorts or staged adoption to compare AI-assisted and non-assisted work of similar complexity, language, repository maturity, and release context. Capture quality-adjusted verified throughput, effort, escaped defects, rework, durability, and DORA delivery performance together. Control for changes in staffing, deployment practices, and task selection. If treatment assignment is not randomized, label the estimates observational and report confounding limits.

Fourth, compare estimated business-impact scores with later observable cost savings, revenue effects, or operational outcomes. Report missing data and uncertainty rather than manufacturing a financial return. Qualitative developer feedback about friction and review load should complement outcome metrics [2].

12 Threats to Validity

Major threats include missing or selective source data, arbitrary task partitioning, complexity inflation, incomplete incident reporting, misleading AI labels, noncomparable teams, subjective business-impact estimates, and low event counts. A higher deployment count can reflect artificial releases; more tests can reflect generated trivial assertions; fewer reported incidents can reflect weaker monitoring.

GSM should counter these risks with evidence thresholds, eligible-event rules, provenance links, immutable classification history, approval of estimates, versioned formulas, outlier review, and confidence levels. Corrections should append a superseding record rather than erase the former one. The dashboard should show denominators and "insufficient evidence" when appropriate. Contributor access and compensation information require role-specific permissions.

13 Implementation Status

The reviewed `main` branch contains a C++ configuration and RocksDB domain-storage foundation, plugin service and lifecycle tests, and OpenAPI contracts for application domains. The main-branch documentation additionally specifies quality/velocity and business-outcome layers. A separate, draft pull request proposes DORA schemas and a service-level API; it explicitly does not implement its runtime. An unmerged architectural decision excludes GSD and coding-agent runtimes from GSM. This paper adopts that exclusion as the intended architecture [9, 8].

Accordingly, diagrams and formulas identify *proposed capabilities*. No empirical lift, validated ranking, deployment-scale benchmark, or finished end-to-end signed-report pipeline is claimed here. The next milestones should validate evidence collection and DORA calculations before assigning credit, forecasting value, or considering incentive decisions.

14 Conclusion

GSM proposes a tool-independent way to evaluate engineering productivity from accepted, verified, and durable results. It distinguishes the health of a team's delivery system from the quality and volume of the work delivered, and from the business value assigned to that work. This separation is especially important when AI reduces implementation effort but leaves architecture, review, reliability, and accountability as central human responsibilities. Its proposed contribution is not a faster coding agent; it is a reproducible method for measuring what engineering work accomplished.

A Minimum Reproducible Measurement Record

A result should be portable across GSM versions without exposing raw private evidence. At minimum, each metric snapshot retains a metric identifier, a profile and formula version, its measurement scope, the closed-open time interval, inclusion and exclusion rules, the event sample identifiers, numerator and denominator (where applicable), any observed durations, an evidence-coverage state, the calculation timestamp, and the reviewer or override record. Recalculation should use the recorded inputs and historical formula rather than silently adopting the currently active policy.

Table 2: Proposed snapshot fields and purpose

Field	Purpose
`scope_ref`	Organization, team, service, or verified workstream
`window`	Start/end instants and reporting zone
`profile_version`	Frozen definitions and eligibility rules
`sample_refs`	Exact input event and evidence identifiers
`counts/durations`	Raw measurements, never only a composite score
`coverage/confidence`	Missing sources, ambiguous correlations, small samples
`review_ref`	Human classifications, decisions, and overrides
`digest`	Repeatable integrity digest of canonical snapshot

B Deployment and Research Reproducibility

The research deployment should record the GSM source revision, relevant OpenAPI contract versions, source-adapter versions, metric-policy revisions, and evidence-ingestion window. Source evidence can remain on the operator-controlled host while sanitized aggregate snapshots are shared for external review. Cross-service comparisons must publish eligibility and exclusion rules, including deployment boundaries, source coverage, sample counts, and the handling of feature flags or coordinated releases.

Reported experimental changes should include the comparison design, allocation of workstreams to conditions, baseline period, task and service characteristics, quality constraints, and confidence intervals. If a period contains no eligible deployments or a workstream lacks verifiable acceptance, it must be excluded or shown as insufficient evidence rather than treated as zero effort or zero defects. These requirements make replication possible while respecting the limits of evidence available from external development systems.

References

[1] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, and J. Butler, "The SPACE of Developer Productivity: There's More to It Than You Think," *ACM Queue*, vol. 19, no. 1, pp. 20–48, 2021. doi:10.1145/3454122.3454124.

[2] A. Noda, M.-A. Storey, N. Forsgren, and M. Greiler, "DevEx: What Actually Drives Productivity?" *Communications of the ACM*, vol. 66, no. 11, pp. 44–49, 2023. doi:10.1145/3610285.

[3] DORA, "DORA's Software Delivery Performance Metrics," https://dora.dev/guides/dora-metrics/, accessed Oct. 2026.

[4] N. Harvey, "A History of DORA's Software Delivery Metrics," DORA, Jan. 2, 2026. https://dora.dev/insights/dora-metrics-history/.

[5] Genius Ventures, "Genius Software Manager: Product Technical Design Specification," repository design specification, 2026.

[6] Genius Ventures, "Quality and Velocity Dashboard," GSM repository design specification, 2026.

[7] Genius Ventures, "Business Performance and Outcome-Based Compensation," GSM repository design specification, 2026.

[8] Genius Ventures, "DORA Software Delivery Performance," proposed design and OpenAPI contract, pull request #10, 2026 (unmerged at time of writing).

[9] Genius Ventures, "Agent Router Boundary," design decision, `docs/agent-router-boundary` branch, 2026 (not merged into `main` at time of writing).