
FROM LOCAL CORRECTNESS TO GLOBAL ENGINEERING COHERENCE:

An Engineering Decision Graph Framework for Long-Horizon Agentic Software Development

Ankush Seth

Greater Toronto Area, Canada
<https://ca.linkedin.com/in/ankushseth>

July 13, 2026

ABSTRACT

Long-horizon coding agents increasingly perform repository-scale implementation, migration, debugging and validation tasks. Sustained execution, however, does not ensure globally coherent engineering. Agents operate under unavoidable contextual incompleteness and therefore infer objectives, select parameters and embed assumptions that may be locally reasonable yet inconsistent with distant dependencies, operational conditions or future scale. This paper introduces the Engineering Decision Graph (EDG), a governance framework that makes those decisions durable, inspectable and revisable. Agentic Engineering Decision Records (AEDRs) represent individual decisions; typed graph relationships connect them to assumptions, evidence, requirements, code, tests, owners, observations and validity conditions. A Decision Engine compiles decision-aware task context, estimates blast radius, detects contradictions and applies policy gates. A retrospective agentic loop re-evaluates decisions as repositories and operating environments change. The framework defines lifecycle states, graph invariants, temporal invalidation triggers, decision-debt diagnostics and an evaluation protocol for repository-scale studies. It operationalizes decision materiality through hard gates and a five-factor index, and represents global coherence as a calibrated multidimensional profile rather than a universal quality score. Reference policies define bounded invalidation propagation, agent quality gates, scalability controls and illustrative governance-burden targets. These mechanisms and thresholds are configurable specifications, not validated universal values. The contribution is a conceptual framework and research agenda; empirical performance claims remain to be tested.

Keywords: agentic coding; engineering decisions; knowledge graphs; software governance; long-horizon agents; retrospective control

The paper makes five contributions:

- A decision-governance formulation of long-horizon coding under unavoidable contextual incompleteness.
- A typed Engineering Decision Graph in which AEDRs, assumptions, evidence, artifacts, tests and operational observations participate in explicit dependency and validity relationships.
- An operational materiality policy and decomposable coherence profile for proportional, calibratable governance.
- A retrospective control loop that searches for contradictions and reopens decisions when their evidence or operating bounds change.
- A practical reference architecture, machine-readable schema and evaluation protocol for studying global engineering coherence.

1. The Long-Horizon Engineering Problem

1.1 From code generation to delegated engineering

Short coding tasks constrain both the objective and the state space. Long-horizon tasks do the opposite. A repository-scale change may require an agent to discover entry points, trace asynchronous consumers, infer invariants, select operational parameters, modify schemas, coordinate migrations, preserve compatibility, and verify behavior across deployment boundaries. Each step introduces decisions that may be technically reasonable in isolation but incompatible in combination.

The risk grows with autonomy. When a human directs each intermediate step, assumptions can be corrected before they harden into code. When an agent operates for hours or days from a broad objective, hidden choices compound. A locally plausible early choice constrains subsequent retrieval, planning and implementation, creating path dependence across the trajectory.

1.2 The context completeness fallacy

A common response is to provide more context. Retrieval, larger context windows, repository maps and persistent memory are valuable, but none creates complete engineering context. Relevant facts can live in code, infrastructure, runtime configuration, production data, incident history, customer contracts, security policies, road maps and human expectations. Some facts are inaccessible; others are stale, contradictory or never documented.

Even if every relevant fact could be assembled, its presence would not ensure appropriate influence on the decision. A coding agent can assign different effective importance to competing instructions, repository patterns, tests and architectural constraints without exposing a stable, reviewable weighting model. A critical global invariant may therefore be available in context yet still be outweighed by a stronger local pattern. Context availability and decision alignment are separate problems.

The objective is not context completeness. It is disciplined decision-making under contextual incompleteness. An agent must distinguish verified facts from inferred assumptions, selected parameters from inherited standards, and current evidence from expectations about the future.

1.3 Four layers of correctness

Layer	Question	Typical evidence
Syntactic	Does the artifact compile, parse or execute?	Compiler, linter, type checker
Behavioral	Does it satisfy the stated behavior?	Unit, integration and acceptance tests
Systemic	Does it preserve wider architectural, security, data and operational invariants?	Architecture tests, dependency analysis, contracts, threat models
Temporal	Will the decision remain suitable as conditions change?	Capacity models, validity clauses, telemetry and review triggers

Conventional coding workflows emphasize syntactic and behavioral correctness. EDG governance adds explicit support for systemic and temporal correctness without treating either as a guarantee.

1.4 Hidden decision surfaces

A request such as “make the workflow reliable” leaves many consequential parameters unspecified. The agent may decide what constitutes failure, which errors are retrievable, how often to retry, where state is persisted, how concurrency is controlled, whether side effects require idempotency, when users are notified, and what evidence is sufficient to declare success. These choices affect reliability, cost, latency, consistency, compliance and future extensibility. They are engineering decisions, even when they appear as small constants or control-flow branches.

PROBLEM STATEMENT

Code review exposes implementation. It does not necessarily expose the objective interpretation, assumptions, rejected alternatives, optimization criteria or conditions under which the chosen design should expire.

2. Requirements for Decision-Aware Autonomy

A governance layer for autonomous coding must satisfy six requirements.

Decision observability. Material choices, assumptions and parameter selections are inspectable without reverse-engineering the diff.

Provenance. Claims and assumptions link to the code, tests, documents, telemetry or human authority that supports them.

Dependency awareness. The system can traverse what a decision depends on, constrains, implements, conflicts with or may invalidate.

Temporal validity. Each decision states the operating conditions and review triggers that bound its expected fitness.

Executable governance. Important assertions connect to tests, policies, monitors, capacity thresholds and approval gates.

Revisability. New evidence can reopen, supersede or retire a decision while preserving history and accountability.

These requirements do not demand that an agent reveal private chain-of-thought. The framework records engineering-facing rationale and evidence: the decision object necessary for review, traceability and control.

3. The EDG Framework

The Engineering Decision Graph is a dynamic, typed representation of engineering knowledge. Decisions, assumptions, evidence, requirements, architectural constraints, tests, operational observations and implementation artifacts are vertices. Semantic relationships are edges. AEDRs create and update decision vertices; the Decision Engine uses graph state to compile context, calculate impact, enforce governance and drive retrospective review.

Engineering Decision Graph framework

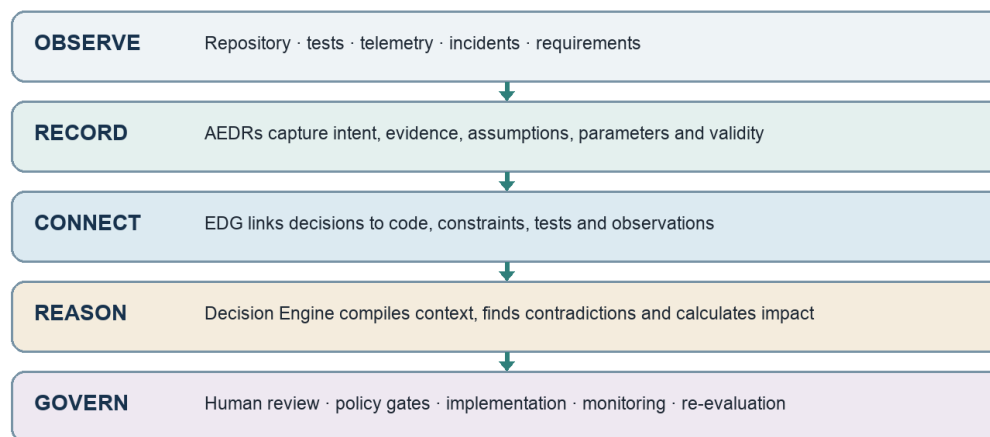


Figure 1. The EDG connects engineering evidence and execution through a durable decision layer.

3.1 Three levels of abstraction

1. Agentic Engineering Decision Record (AEDR): the atomic record of one decision.
2. Engineering Decision Graph (EDG): the dependency and evidence model that connects records to the system.
3. Engineering Decision Engine (EDE): the active control layer that queries, validates and evolves the graph.

3.2 Decision lifecycle

Every material decision moves through explicit states: draft, challenged, accepted, active, at-risk, reopened, superseded or retired. State transitions require evidence and preserve history. A code change can be merged only when the decisions that authorize it satisfy the applicable policy gate; low-risk work can use lighter records, while high-impact decisions require richer evidence and human approval.

3.3 Scope discipline

Not every edit merits a full record. Formatting, mechanical refactors and generated changes may inherit an existing decision. The framework therefore classifies decision materiality before requiring an AEDR.

3.4 Operational materiality test

A full AEDR is mandatory when a decision crosses any hard gate: it changes a contract used outside the component; changes a security, privacy, compliance, tenant or financial boundary; creates an irreversible data mutation, destructive migration or external side effect; or establishes a shared architectural dependency, consistency model or data-lifecycle rule.

Decisions outside those gates use a default five-factor materiality index. Each factor is scored 0, 1 or 2 using the following anchors.

Factor	0 - Low	1 - Moderate	2 - High
Blast radius	One local artifact	Several artifacts in one component	Multiple components, services or external consumers
Reversibility	Mechanical rollback	Rollback needs coordination or migration	Irreversible, destructive or contract-breaking
Uncertainty	Established standard with direct evidence	Some inference or incomplete evidence	Unsupported premise or disputed evidence
Longevity	Ephemeral or generated	Expected to persist for one release cycle	Establishes a durable constraint or precedent
Operational impact	No meaningful effect	Bounded effect on cost, latency or reliability	Material SLO, capacity, cost, safety or compliance effect

$$M = B + R + U + L + O \quad (\text{each factor: } 0..2)$$

The baseline policy requires an AEDR when M is 5 or greater. Scores of 5-6 permit a lightweight AEDR; scores of 7-10 require the full record and challenge pass. A score below 5 may inherit an active decision, but the classifier result remains in change metadata. Hard gates always override the score. Organizations may recalibrate thresholds using incidents, review reversals and false escalation rates, but must version the policy and may not lower a classification silently.

4. Agentic Engineering Decision Records

An AEDR is a human-readable and machine-readable decision artifact created alongside a material code change. It extends the spirit of Architecture Decision Records from architecturally significant human choices to the broader decision surface of autonomous implementation. The AEDR does not store private reasoning traces; it stores the engineering claims required to evaluate and govern the decision.

AEDR field	Required content
Identity and status	Unique identifier, title, lifecycle state, owner, creation date and supersession links.
Intended outcome	Observable result the change must create and harms it must avoid.
Objective interpretation	How the request was translated into an engineering problem and optimization boundary.
System boundary	Components, data stores, consumers, external systems and environments considered.
Evidence and provenance	Repository locations, tests, telemetry, policies, incidents and human decisions supporting the record.
Assumptions	Claims treated as true without complete verification, with confidence and owner.
Selected parameters	Timeouts, retry limits, capacities, algorithms, thresholds and defaults.
Alternatives and trade-offs	Plausible options, rejection rationale and objective weights.
Invariants	System conditions that the implementation must preserve.
Temporal horizon	Expected workload, topology, scale, policy regime, team capability and review date.
Invalidation conditions	Events or evidence that move the decision to at-risk or reopened.
Validation plan	Tests, measurements, monitors and approvals linked to acceptance.

4.1 Evidence classes

Evidence should be typed because not all support is equivalent. Direct executable evidence includes tests and policy checks. Observational evidence includes traces, metrics and incidents. Documentary evidence includes requirements, contracts and architectural standards. Pattern evidence derives from existing repository practice. Human authority records explicit stakeholder or reviewer decisions. Unsupported assumptions remain visible and accumulate decision debt until resolved.

4.2 Confidence without false precision

Numeric confidence can imply a measurement that does not exist. AEDRs therefore use calibrated support bands—verified, strongly supported, weakly supported and unknown—unless the organization has a validated probabilistic model. Each band must link to evidence and a review action. “Unknown” is an actionable state, not a documentation failure.

4.3 Temporal validity clause

VALIDITY CLAUSE TEMPLATE

This decision is valid while [workload], [topology], [dependency behavior], [policy regime] and [service objective] remain within the recorded bounds. It is automatically reopened when any bound is crossed, evidence freshness expires, a dependent decision changes, or an incident contradicts the expected outcome.

5. Engineering Decision Graph Model

5.1 Node types

Node type	Examples	Governance role
Decision	Retry policy, cache strategy, consistency model	Authorizes implementation and records trade-offs
Assumption	Provider honors idempotency keys	Exposes uncertain premises
Requirement invariant /	No duplicate external effects	Constrains acceptable solutions
Evidence	Test, benchmark, incident, policy, telemetry	Supports or contradicts claims
Artifact	Function, service, schema, config, deployment	Binds decisions to implementation
Observation	Latency, error rate, saturation, security event	Tests expected outcomes over time
Actor / owner	Team, reviewer, service owner	Assigns authority and accountability
Trigger	Scale threshold, dependency upgrade, review date	Reopens or retires decisions

5.2 Typed edges

Relationship	Meaning
depends_on	The source decision requires the target decision or premise to remain valid.
assumes	The source relies on a claim that is not fully verified.
supports / contradicts	Evidence increases or decreases support for a decision or assumption.
implements	A code, configuration or infrastructure artifact realizes a decision.
constrained_by	A requirement, policy or invariant limits the decision space.
verified_by / monitored_by	A test, check or observation evaluates the decision.
conflicts_with	Two active decisions cannot both remain coherent under the same scope.
supersedes	A newer decision replaces an earlier decision while preserving lineage.
invalidates / reopened_by	A change or observation breaks a validity condition.
owned_by	An actor is accountable for review and disposition.

5.3 Graph invariants

- Every active material code path is governed by at least one active decision or an explicit inherited policy.
- Every active decision has an owner, scope, evidence set, validation link and validity clause.
- A superseded decision cannot authorize new implementation, but remains queryable for history.
- Contradictory active decisions within overlapping scope require an explicit resolution edge.
- An invalidated premise propagates an at-risk state through dependent decisions according to policy.
- Security, privacy, financial and irreversible side-effect decisions require human authority before activation.

5.4 Decision-aware context compilation

Before implementation, the engine starts from task terms and touched artifacts, expands callers and consumers, follows relevant decision and invariant edges, retrieves fresh evidence, and prunes the result to a task-specific dossier. The dossier includes inherited decisions, high-centrality assumptions, unresolved conflicts, system boundaries, invalidation conditions and required validation. The graph does not replace repository exploration; it directs and records it.

5.5 Invalidation propagation protocol

Invalidation is processed as a versioned review event rather than an unbounded graph broadcast. An event identifies the changed node, evidence revision, trigger class, affected scope and originating decision version. Events with the same decision, version and trigger class are coalesced within a policy-defined window so overlapping operational, structural and temporal signals do not create duplicate reviews.

Edge class	Examples	Default propagation
Hard dependency	depends_on, assumes, constrained_by	Traverse to dependents and mark them at-risk when the premise or constraint no longer holds.
Implementation	implements	Re-evaluate the governing decision when a bound artifact changes materially.
Evidence	supports, contradicts, verified_by, monitored_by	Recalculate support and validity before changing decision state.
Conflict / lineage	conflicts_with, supersedes, invalidates	Apply explicit conflict, supersession or invalidation policy.
Informational	owned_by and descriptive links	Route or enrich review without propagating risk state.

Propagation uses a visited set and collapses strongly connected components so cycles are evaluated as one decision group. Traversal is bounded by edge policy, decision scope, risk tier, maximum depth and review budget. A high-risk path that reaches a boundary is escalated; it is never treated as safely exhausted. The default state transition is active to at-risk. A decision moves to reopened only when a recorded validity condition is violated, contradictory evidence satisfies a policy threshold, or a hard gate requires renewed authority.

$$P(d) = \text{Risk}(d) * \text{Centrality}(d) * \text{TriggerSeverity}(d)$$

The work queue processes the highest-priority affected decisions first and terminates when no new state or evidence revision is produced. Conflict resolution follows explicit precedence: safety, security, compliance and irreversible-action policies; system invariants; accepted decisions within matching scope; and explicit supersession. Remaining incompatible decisions require recorded human disposition rather than automatic recency-based resolution.

6. Retrospective Agentic Control Loop

The retrospective loop turns the EDG from documentation into a control system. It separates solution formation from contradiction search so the same reasoning path does not silently create and approve its own assumptions.

Retrospective agentic control loop

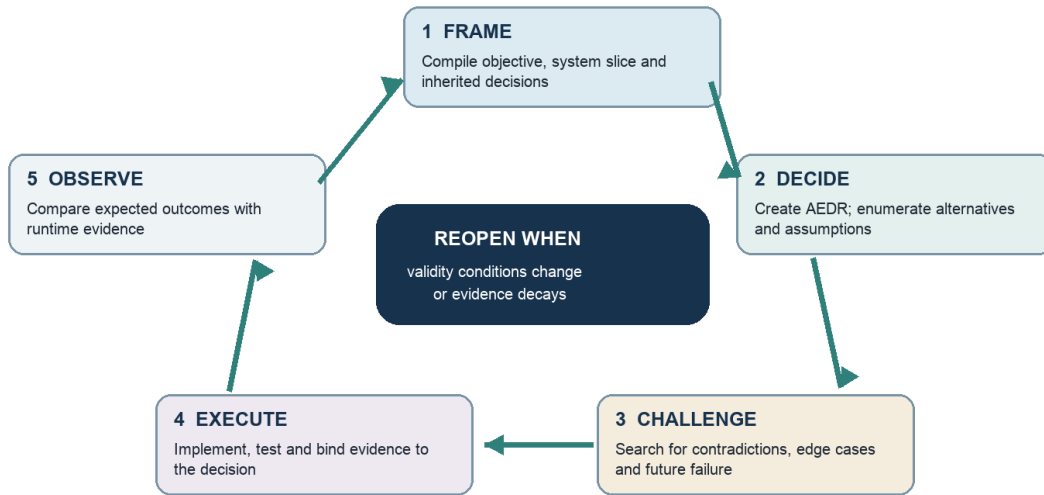


Figure 2. Decisions are reopened when operating evidence or validity conditions change.

Frame. Compile the interpreted objective, relevant system slice, prior decisions, constraints and unknowns.

Decide. Create a provisional AEDR with alternatives, selected parameters, trade-offs and expected outcomes.

Challenge. Assume the decision is wrong. Search indirect callers, alternate entry points, background work, permissions, data lifecycle, migration paths, rollback behavior and future-scale contradictions.

Execute. Implement only after contradictions are resolved or explicitly accepted; bind the diff and validation evidence to the decision.

Observe. Compare runtime outcomes with the AEDR, update evidence freshness and reopen decisions when triggers fire.

6.1 Trigger classes

Trigger class	Examples	Default response
Structural	New caller, schema migration, region expansion, dependency replacement	Recalculate blast radius and contradiction set
Operational	SLO breach, cost increase, queue saturation, incident	Compare observed and expected outcomes
Temporal	Review date, evidence expiry, policy renewal	Refresh evidence or mark decision at-risk
Organizational	Owner change, new compliance regime, team capability change	Reassign authority and reassess constraints
Product	New requirement, changed customer contract, new tenant model	Recompile objective and dependent decisions

6.2 Human control points

Autonomy is bounded by decision class. The engine may autonomously accept low-risk records that satisfy policy and executable evidence. Human review remains mandatory for irreversible actions, material security or privacy changes, regulatory interpretation, externally visible contracts, major cost commitments, destructive migrations, and unresolved high-centrality assumptions. The graph makes escalation criteria explicit instead of relying on ad hoc agent judgment.

7. Decision Engine and Reference Architecture

The Decision Engine integrates with the software delivery toolchain. Its components can be implemented incrementally and need not share a single database or agent runtime.

Component	Responsibility	Primary outputs
Repository indexer	Parses code, schemas, configuration, infrastructure, tests and ownership metadata.	Artifact graph and change events
Evidence adapters	Ingest telemetry, incidents, tickets, policies and human approvals.	Typed, time-stamped evidence nodes
AEDR service	Creates, validates, versions and signs decision records.	Decision nodes and lifecycle events
Graph store	Maintains typed edges, scopes, lineage and traversal indexes.	Queryable decision knowledge model
Context compiler	Builds the minimum sufficient decision-aware dossier for a task.	Task context package
Impact analyzer	Calculates dependency paths, centrality, conflicts and blast radius.	Impact report and review routing
Policy engine	Applies risk tiers, approval rules, evidence requirements and merge gates.	Pass, block or escalate decision
Retrospective scheduler	Listens for triggers and initiates targeted re-evaluation.	Reopened records and review queues

7.1 End-to-end workflow

1. A task or change event identifies the initial artifact and objective scope.
2. The context compiler retrieves related decisions, constraints, assumptions, tests, owners and evidence.
3. The implementation agent creates a provisional AEDR and code plan.
4. A challenge pass searches the repository and graph for contradicting evidence and omitted impact.
5. The policy engine applies proportional gates and routes required approvals.
6. Implementation and validation bind code, tests and results to the accepted decision.
7. Deployment observations update expected-outcome evidence and trigger retrospective review when necessary.

7.2 Storage and interoperability

A practical implementation stores AEDRs as version-controlled YAML or JSON for portability while projecting nodes and edges into a graph database or relational graph layer for traversal. Stable identifiers allow links from commits, pull requests, tests, dashboards and incident systems. The canonical record remains reviewable without specialized infrastructure; the graph projection provides operational intelligence.

7.3 Security and integrity

Decision records can contain sensitive architecture, incidents and business constraints. Access control must follow artifact scope and tenant boundaries. Provenance events should be append-only, tamper-evident and attributable. Agent-written evidence must be distinguishable from human authority. Automated agents must not fabricate approval, downgrade risk, or silently delete contradictory evidence.

7.4 Agent capability and quality gates

The framework treats an agent as a candidate decision author and reviewer, not as a reliable authority by default. Deployment requires observable quality gates between agent output and decision acceptance.

Capability	Operational measure	Escalation condition
Material-decision detection	Precision and recall against human-labeled decisions	Missed hard gate, low recall or unstable classification
AEDR generation	Required-field completeness and evidence-attribution accuracy	Missing provenance, unsupported claim or inconsistent scope
Contradiction search	Discovery rate on seeded and historical contradictions	High-risk scope with insufficient independent findings
Reopening judgment	False-reopening and missed-reopening rates	Threshold breach or unresolved validity conflict
Decision authority	Human override and reversal frequency	Authority required by policy or repeated agent reversal

When evidence coverage, classifier stability or contradiction-search quality falls below the applicable gate, the agent must abstain or escalate. Repetition, self-consistency or confident language cannot substitute for missing evidence. Agent capability metrics are versioned by model, prompt, toolset, repository domain and policy configuration because performance in one setting does not establish reliability in another.

7.5 Scalability and maintenance strategy

- Ingest repository and operational changes incrementally instead of rebuilding the graph for every task.
- Partition nodes by service, domain, tenant and authority boundary while preserving explicit cross-partition edges.
- Coalesce trigger events and cache traversal results with dependency-aware invalidation.
- Expire or refresh evidence according to typed half-lives; archive superseded decisions without deleting lineage.
- Use lazy recomputation for low-centrality regions and eager review for hard-gate or high-centrality paths.
- Monitor graph size, traversal latency, stale-edge rate, orphaned decisions, event backlog and review-queue age.

These mechanisms bound routine cost but do not guarantee enterprise-scale performance. Scale claims require measurement on repositories and decision graphs with representative topology, event rates and organizational boundaries.

8. Decision Analytics and Coherence Metrics

EDG metrics are diagnostic signals, not substitutes for judgment. They help prioritize review and detect governance drift.

Metric	Operational interpretation
Decision centrality	Weighted number and importance of reachable dependents. High centrality indicates architectural leverage and review priority.
Blast radius	Decisions, artifacts, tests, owners and operational controls reachable through impact-relevant edges within policy depth.
Evidence freshness	Age-adjusted support score based on evidence type, expected half-life and last verification time.
Decision entropy	Dispersion of support across alternatives, contradictions and unresolved assumptions. Higher entropy indicates weaker convergence.
Decision debt	Risk-weighted sum of unsupported assumptions, expired evidence, unresolved conflicts and missed review triggers.
Coverage	Share of material changed artifacts linked to active decisions, invariants and validation evidence.
Global coherence profile	Decomposable view of decision alignment, evidence, traceability, operations and validity, with contradiction and debt risks shown separately.

8.1 Global coherence profile

Global engineering coherence is represented first as a profile rather than a universal scalar. The profile preserves dimensions that can move independently and require different interventions.

$$\text{GCP} = (A, E, T, O, V) \quad \text{risks} = (K, D)$$

A is alignment among active decisions; E is evidence coverage and freshness; T is traceability to requirements, code, tests and controls; O is conformance between expected and observed operational behavior; V is the validity of assumptions and temporal conditions. K reports unresolved contradiction risk and D reports decision debt. Unknown or unavailable evidence remains unknown rather than being converted to a neutral or favorable value.

A deployment may calculate a local weighted summary for routing and trend analysis, but the profile remains the authoritative representation.

$$CW = wA*A + wE*E + wT*T + wO*O + wV*V - wK*K - wD*D$$

The summary is a policy-specific diagnostic, not proof of correctness and not a basis for comparison across organizations. Reviewers must be able to inspect every component, weight, evidence source and contributing record.

8.2 Calibration protocol

Each deployment defines component scales, evidence half-lives, minimum coverage and hard gates before using the profile operationally. Initial thresholds are baselined on historical incidents, escaped regressions, decision reversals and review outcomes. Teams then measure false-positive and false-negative escalations, run sensitivity tests on weights and thresholds, and version every calibration change. A profile is not considered calibrated until its components show a documented relationship to the outcomes they are intended to predict or prevent.

8.3 Anti-gaming requirements

- Coverage cannot increase merely by creating low-information records.
- Agent-generated evidence cannot satisfy human-approval requirements.
- Unknowns remain visible; they cannot be converted to high confidence through repetition.
- Metrics expose raw components and provenance, not only aggregate scores.
- A favorable score cannot override a hard policy gate or known safety violation.

9. Worked Example: Resilient Tool Execution

9.1 Task

An agent receives the objective: “Add retries to external tool execution so transient failures do not fail the workflow.” A local implementation could add three exponential-backoff attempts around the provider call. The EDG process widens the decision boundary before code is changed.

9.2 Provisional AEDR

Field	Recorded decision
Outcome	Increase completion during transient provider failures without duplicating external effects or violating latency objectives.
Parameters	Three attempts; exponential backoff with jitter; 30-second maximum delay; retry only timeouts and selected 5xx responses.
Idempotency	Use workflow-step identifier plus connector version as the idempotency key.
Persistence	Commit attempt state durably before worker rescheduling; recover across process failure.
Assumptions	Step identifiers are globally unique; connectors forward the key; provider behavior is consistent across regions.
Alternatives	Immediate retry rejected for saturation risk; unlimited retry rejected for cost and queue starvation; process-local state rejected for crash loss.
Validity	Single-region topology, current provider contracts, tenfold workload ceiling, user-visible completion objective under two minutes.
Triggers	Active-active rollout, non-idempotent connector, provider contract change, duplicate-side-effect incident, latency SLO breach.

9.3 Graph expansion

The decision depends on durable checkpointing and idempotent connector behavior; it is constrained by the user latency objective and provider rate limits; it is implemented by the executor, queue configuration and checkpoint repository; and it is verified by crash-recovery, concurrent-worker and duplicate-execution tests. Monitoring links include retry rate, duplicate-effect detections, queue age and provider throttling.

9.4 Contradiction search

The challenge pass discovers that one legacy connector ignores idempotency keys, a cancellation path can race with delayed retries, and the current checkpoint write occurs after the external side effect. These findings invalidate the naive wrapper. The accepted design introduces a connector capability declaration, prevents automatic retries for non-idempotent connectors, persists an execution intent before dispatch, adds a reconciliation path for ambiguous outcomes, and makes cancellation a terminal gate checked before every attempt.

9.5 Retrospective behavior

Six months later, an active-active regional rollout changes the uniqueness and ordering assumptions. The deployment-topology event follows an invalidates edge to the retry AEDR, moves it to at-risk, identifies dependent connectors and tests, and opens a review. The system does not assume that passing legacy tests proves continued fitness. It asks whether the original decision remains valid under the new topology.

RESULT

The value of the EDG is not that it chooses “three retries.” It makes the conditions, dependencies and consequences of that choice visible before implementation and revisitable after deployment.

10. Operating Model and Adoption

10.1 Roles

Role	Responsibility
Explorer	Maps affected artifacts, prior decisions, dependencies and unknowns.
Implementer	Creates the provisional AEDR and implementation.
Challenger	Searches for contradictory evidence, hidden consumers and future failure modes.
Validator	Binds tests, checks and measurements to decision assertions.
Decision owner	Accepts accountability for validity, review and disposition.
Policy owner	Defines risk tiers, evidence standards and human control points.

These roles can be performed by separate agents, separate passes, humans, or a hybrid. Separation is logical rather than necessarily organizational: a small team can use independent sessions, separately compiled context and targeted human sampling without assigning one person to every role. The record must disclose when the same person, model or context performed multiple roles so reviewers can assess correlated failure risk.

10.2 Risk tiers

Tier	Change profile	Minimum control
T0	Mechanical or formatting change; no semantic impact	Inherited decision and standard tests
T1	Reversible local behavior with bounded impact	Compact AEDR, automated challenge and validation
T2	Cross-component contract, data, cost or operational impact	Full AEDR, owner review and deployment observation
T3	Security, privacy, compliance, irreversible migration or external side effect	Human authority, independent challenge, rollback plan and monitored release

10.3 Adoption path

1. Start with a narrow decision class—retries, caching, public APIs or database migrations—where assumptions and invalidation triggers are visible.
2. Define the minimum AEDR schema, identifiers and lifecycle states in version control.
3. Link records to pull requests, code ownership, tests and operational dashboards.
4. Add graph projection and blast-radius traversal after record quality stabilizes.
5. Introduce trigger-driven retrospective review for incidents, dependency changes and scale thresholds.
6. Calibrate metrics and policy gates using observed review value, false positives and escaped defects.

10.4 Failure modes of the framework

- Record theater: verbose AEDRs that restate the diff without exposing decisions.
- Graph drift: edges become stale because code and operational changes are not ingested.
- False authority: agent-generated confidence is mistaken for verified evidence.
- Alert fatigue: broad triggers reopen too many low-impact decisions.
- Governance capture: teams optimize metrics rather than engineering outcomes.

- Excess ceremony: low-risk work receives controls disproportionate to its blast radius.

10.5 Governance burden model

Governance burden includes authoring, review, retrospective processing, graph maintenance and operational integration. A deployment can estimate total burden with a decomposable model rather than treating record count as the cost proxy.

$$B = Na * Ta + Nr * Tr + Nt * Tt + Cg + Co$$

Na, Nr and Nt are the numbers of authored records, reviews and retrospective evaluations; Ta, Tr and Tt are their median handling times; Cg is graph-maintenance cost; and Co is operational-integration cost. Teams should report both human time and compute consumption.

The following ranges are illustrative planning targets for a pilot, not measured performance claims or universal service levels.

Tier	Illustrative incremental burden	Control profile
T0	Less than 1 minute, primarily automated	Classification metadata, inherited decision and standard tests
T1	5-15 minutes	Compact AEDR, automated challenge and validation
T2	20-60 minutes	Full AEDR, owner review and deployment observation
T3	60-240 minutes plus release monitoring	Independent challenge, human authority, rollback planning and monitored release

A deployment should suspend or simplify a control when its measured review value does not justify its burden, except where a hard safety, security, compliance or irreversible-action gate applies. Empirical studies must test these targets and report distribution tails, not only averages.

10.6 Small-team deployment profile

- Use one implementation pass and one context-independent challenge pass rather than requiring separate teams.
- Require human review for T3 decisions and disputed T2 decisions; sample lower tiers based on observed error rates.
- Prefer repository-native AEDR files and scheduled batch retrospectives before introducing dedicated graph infrastructure.
- Record role overlap, model identity, context source and human overrides so independence remains auditable.
- Expand controls only when incidents, reversals or measured decision risk justify the additional burden.

11. Evaluation Framework

The framework should be evaluated as a software-engineering governance intervention, not only as a prompting technique. The central question is whether EDG-governed agents produce changes that remain more coherent under hidden dependencies, altered operating conditions and subsequent maintenance.

11.1 Experimental design

A controlled study compares a standard coding-agent workflow with an EDG-governed workflow on repository-scale tasks. Both conditions receive the same model, tools, initial task statement and compute budget. The EDG condition adds decision-aware context compilation, AEDR creation, an independent contradiction pass and validity-trigger evaluation.

11.2 Task construction

- Requirements distributed across distant modules, configuration and tests.
- Hidden consumers or asynchronous paths not named in the issue.
- Parameter choices that trade reliability, cost and latency.

- Future-state perturbations such as tenfold load, multi-region deployment or dependency behavior change.
- Operational evidence that contradicts an initially reasonable assumption.
- Follow-on maintenance tasks performed in a later, context-independent agent session.

11.3 Outcomes

Dimension	Measures
Immediate quality	Test pass rate, hidden-test performance, patch validity, regression count
Decision quality	Assumptions found, alternatives considered, parameter justification, evidence precision, materiality-classifier precision and recall
System coverage	Affected components, consumers, invariants and operational controls discovered
Temporal robustness	Performance after scale, topology, policy or dependency perturbation
Review efficiency	Human time, useful findings, escalation precision, false-positive burden
Lifecycle value	Decision reuse, incident diagnosis time, maintenance success, reopened-decision accuracy
Propagation quality	Relevant invalidations found, false propagation, missed dependents, traversal cost and time to disposition
Governance cost	Latency, human time, token or compute cost, record volume, graph-maintenance effort and burden tails

11.4 Research hypotheses

- H1: EDG governance increases discovery of unstated assumptions and affected components.
- H2: EDG-governed patches show fewer systemic regressions under hidden and future-state tests.
- H3: Typed validity triggers identify expired decisions earlier than periodic documentation review.
- H4: Decision-aware context improves later-session maintenance by preserving rationale and constraints.
- H5: Benefits concentrate in high-coupling, high-uncertainty and high-blast-radius tasks; low-risk tasks require lighter controls.
- H6: Risk-bounded invalidation propagation identifies consequential dependent decisions with lower review burden than unrestricted traversal.
- H7: A context-independent challenge pass discovers more material contradictions than a single agent pass that creates and approves the same decision.

11.5 Required capability reporting

Evaluations must report material-decision detection precision and recall, AEDR completeness, evidence-attribution accuracy, contradiction-discovery rate, false and missed reopenings, human override frequency, review burden and graph-maintenance cost. Results must be stratified by model, repository domain, risk tier and task coupling. A passing implementation benchmark does not establish decision-governance capability, and aggregate averages must not conceal hard-gate failures.

12. Limitations and Threats to Validity

The EDG cannot guarantee complete context or globally optimal engineering. It formalizes what is known, assumed and monitored; undocumented human intent can still remain absent. Static analysis misses dynamic behavior. Telemetry can be incomplete or misleading. Graph edges can decay. Agents can produce plausible but inaccurate rationale. Human reviewers can approve weak decisions. Future conditions can exceed every recorded scenario.

The framework also creates cost. Record authoring, graph maintenance, evidence ingestion and retrospective review consume compute and human attention. Benefits must be evaluated against this overhead. Risk-based proportionality is therefore part of the framework, not an optional optimization. The burden ranges, propagation rules, materiality thresholds and quality gates in this paper are reference policies and planning targets; they are not validated universal values.

Research evaluations face contamination and construct-validity risks. Passing tests does not equal systemic correctness; materiality thresholds and coherence dimensions can encode organizational preferences; benchmark repositories may not represent enterprise architecture; and agent behavior changes with tools, prompts and model versions. A materiality index can over-document routine work or miss consequential choices, while an aggregate coherence summary can conceal a weak component. Comparative studies must report the full harness, classifier policy, component definitions and calibration data, and must conduct ablations of records, graph retrieval, challenge passes and triggers.

13. Related Work and Differentiation

Architecture Decision Records preserve the context and consequences of architecturally significant choices in small, durable artifacts [1]. Earlier work on architecture decisions treated decisions as first-class design knowledge and emphasized their relationships and rationale [2]. The EDG extends these traditions to autonomous implementation: records are agent-generated under policy, cover intermediate engineering choices, connect directly to code and operations, and participate in automated retrospective control.

Agent frameworks such as ReAct interleave reasoning and action [3], while Reflexion uses feedback and episodic reflection to improve later attempts [4]. These approaches improve task execution. The EDG addresses a different layer: durable, inspectable engineering governance across tasks, agents and time. The unit of memory is not only a trajectory reflection; it is a typed decision with dependency, evidence, authority and validity semantics.

Repository-level benchmarks such as SWE-bench demonstrate that real software issues require navigation and multi-file reasoning beyond self-contained coding problems [5]. Testing-agent work shows that generated tests can improve filtering of candidate fixes [6]. The EDG treats tests as necessary evidence nodes while explicitly representing the requirements, assumptions and future conditions that tests may omit.

Retrieval-augmented generation connects model output to external knowledge [7]. Decision-aware context compilation is a specialized retrieval problem, but its query is governed by decision dependency, risk and validity—not semantic similarity alone. Technical-debt research also establishes that expedient choices create future obligations [8]. Decision debt narrows that lens to unsupported or decaying decision premises in autonomous engineering.

DIFFERENTIATION

The framework does not claim to invent records, knowledge graphs, reflection, retrieval or policy engines individually. Its contribution is their integration into a lifecycle control system for autonomous coding, with AEDRs as atomic nodes, typed engineering dependencies, executable validity conditions and retrospective re-evaluation.

14. Conclusion

The next step in agentic software development is not merely longer execution. It is accountable execution. Long-horizon coding agents inevitably act with incomplete context, infer objectives, select parameters and make trade-offs. When those decisions remain hidden, a technically competent trajectory can produce code that is locally correct yet systemically or temporally incoherent.

The Engineering Decision Graph provides a durable control layer. AEDRs make engineering decisions observable. Typed graph relationships expose dependencies and blast radius. The Decision Engine compiles relevant context and enforces proportional governance. Retrospective loops compare past decisions with new code and operational evidence, reopening them when their premises expire.

This changes the artifact of autonomous coding from a patch plus a test report into a traceable chain of intent, evidence, assumptions, authorization, implementation and observation. The objective is not certainty. It is a system that knows what it decided, why it decided it, what the decision depends on, and when that decision should no longer be trusted.

The operational mechanisms in this paper provide a reference specification, not a claim of validated universal performance. Invalidation policies, agent quality gates, materiality thresholds, coherence calibration and burden targets must be tested and adapted against real repositories, teams and operating conditions. The

framework defines what should be observable and governable; empirical work must establish how reliably and economically agents can perform those functions.

Appendix A. AEDR Machine-Readable Schema

The following illustrative YAML shows the minimum portable representation. Implementations can extend it with domain-specific controls while preserving stable identifiers and lifecycle semantics.

```
aedr_version: 1.0
id: AEDR-2026-0042
title: Resilient external tool execution
status: active
owner: workflow-platform
materiality:
  hard_gate: external_side_effect
  factors: {blast_radius: 2, reversibility: 1, uncertainty: 2, longevity: 2,
operational_impact: 2}
  score: 9
  record_class: full
scope:
  artifacts: [tool_executor, checkpoint_repository, queue_policy]
objective:
  outcome: Recover from transient failures without duplicate side effects
  optimization: [reliability, correctness, latency, cost]
decisions:
  retry_attempts: 3
  backoff: exponential_with_jitter
  max_delay_seconds: 30
assumptions:
  - id: A-1
    claim: Connectors preserve idempotency keys
    support: weakly_supported
    owner: integrations
evidence:
  - id: E-1
    type: integration_test
    uri: tests/tool_execution/test_idempotency.py
validity:
  conditions: [single_region, provider_contract_v3, load_lt_10x]
  review_by: 2027-01-31
  triggers: [topology_change, provider_contract_change, duplicate_incident]
propagation:
  policy_version: edg-propagation-1
  max_depth: 4
  review_budget: 50
  boundary_action: escalate
relationships:
  - {type: depends_on, target: AEDR-2025-0018}
  - {type: constrained_by, target: INV-NO-DUPLICATE-EFFECTS}
  - {type: verified_by, target: E-1}
validation:
  required: [crash_recovery, concurrent_workers, cancellation_race]
approvals:
  - {role: service_owner, status: accepted}
```

Appendix B. Representative EDG Queries

Query class	Representative question
Impact	Which active decisions, tests, monitors and owners are reachable if AEDR-2026-0042 changes?
Contradiction	Which evidence contradicts assumptions inherited by artifacts in this pull request?
Validity	Which active decisions have expired evidence or violated operating bounds?
Coverage	Which changed public interfaces lack an active governing decision and contract test?
Lineage	Why does this service use Redis, and which decisions would be reopened if it were removed?
Security	Which new callers reach a security-sensitive function, and which decisions authorize them?
Temporal	Which decisions were scoped to a single region but now govern multi-region artifacts?
Debt	Which high-centrality decisions depend on unsupported assumptions?

Appendix C. Glossary

Term	Definition
Agentic Engineering Decision Record (AEDR)	A human- and machine-readable record of one material engineering decision, including its objective, evidence, assumptions, parameters, trade-offs, validity conditions and validation.
Engineering Decision Graph (EDG)	The typed graph that connects engineering decisions to assumptions, requirements, evidence, implementation artifacts, tests, observations, owners and triggers.
Engineering Decision Engine (EDE)	The active control layer that compiles decision-aware context, analyzes impact and contradictions, applies policy gates and initiates retrospective review.
Local correctness	Conformance to the explicit task and available tests within the examined implementation scope.
Systemic correctness	Preservation of architectural, security, data, integration and operational invariants across the wider system.
Temporal correctness	Continued suitability of a decision within its stated time, scale, topology, workload, dependency and policy horizon.
Global engineering coherence	Consistency among decisions, architecture, implementation, tests, operations and organizational intent.
Retrospective agentic loop	A recurring control process that frames, decides, challenges, executes and observes, reopening decisions when evidence or validity conditions change.
Decision-aware context compilation	Construction of a task-specific engineering dossier by traversing relevant decisions, dependencies, constraints, evidence, owners and affected artifacts.
Decision blast radius	The reachable set of decisions, artifacts, tests, monitors and owners potentially affected when a decision or premise changes.
Decision centrality	A measure of a decision’s architectural leverage based on the number and importance of its reachable dependents.
Decision entropy	A diagnostic measure of uncertainty arising from competing alternatives, contradictory evidence and unresolved assumptions.
Decision debt	Risk-weighted exposure created by unsupported assumptions, stale evidence, unresolved conflicts and expired validity conditions.
Evidence freshness	The degree to which supporting evidence remains current relative to its type, expected half-life and last verification date.
Temporal validity clause	The explicit operating bounds within which a decision is expected to remain appropriate.
Invalidation trigger	A structural, operational, temporal, organizational or product event that moves a decision to at-risk or reopened status.
Invalidation propagation protocol	The bounded, typed-edge process that coalesces trigger events, identifies dependent decisions, changes risk state and escalates unresolved traversal boundaries.
Decision materiality index	A five-factor classification of blast radius, reversibility, uncertainty, longevity and operational impact used with hard gates to determine AEDR requirements.
Global coherence profile	A decomposable diagnostic of decision alignment, evidence, traceability, operational conformance and validity, with contradiction risk and decision debt reported separately.
Governance burden	Human and computational cost attributable to decision authoring, review, retrospective evaluation, graph maintenance and operational integration.

References

- [1] M. Nygard, “Documenting Architecture Decisions,” Cognitect, 2011.
<https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions>
- [2] J. Tyree and A. Akerman, “Architecture Decisions: Demystifying Architecture,” IEEE Software, vol. 22, no. 2, pp. 19–27, 2005. doi:10.1109/MS.2005.27.
- [3] S. Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” International Conference on Learning Representations (ICLR), 2023. https://openreview.net/forum?id=WE_vluYUL-X.
- [4] N. Shinn et al., “Reflexion: Language Agents with Verbal Reinforcement Learning,” Advances in Neural Information Processing Systems, vol. 36, 2023. arXiv:2303.11366.
- [5] C. E. Jimenez et al., “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” International Conference on Learning Representations, 2024.

- [6] N. Mündler, M. N. Mueller, J. He, and M. Vechev, “SWT-Bench: Testing and Validating Real-World Bug-Fixes with Code Agents,” *Advances in Neural Information Processing Systems*, vol. 37, 2024. doi:10.52202/079017-2601.
- [7] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [8] P. Kruchten, R. L. Nord, and I. Ozkaya, “Technical Debt: From Metaphor to Theory and Practice,” *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012. doi:10.1109/MS.2012.167.