

GOVERNED AUTONOMY

Engineering Methodology

A paper in the Governed Autonomy collection

Andrew Durden
Independent research
Publication edition 2026.10.7

Research on authority, evidence, and continuity in autonomous systems.

Content identifier: 30f2410373ee5d9211c2ab5e

Contents

Engineering Methodology	3
Responsibility and the subject of evidence	3
A test can change the claim	4
Working with substantive AI proposals	4
Integration, operations and continuing responsibility	5
Credit and novelty	5
Acceptance and review	5

Engineering Methodology

Andrew Durden is the sole human engineer and architect behind Cuttlefish and Freehold. His responsibility includes technical conception, architecture, implementation, integration, production infrastructure and operation. For Cuttlefish this includes the enterprise backend, GitOps, production deployments, Azure and AWS engineering and orchestration, and the Azure-to-AWS migration. These are owner-attested attribution statements. They do not assert that every mechanism discussed in the research is deployed.

AI systems contributed substantively to exploration, research, design, implementation, debugging, testing, critique, validation and engineering execution. Some architectural concepts originated with Durden; discoveries and refinements also emerged through AI collaboration. Durden directed the work and retained final technical authority, integration decisions, engineering acceptance and production responsibility. Describing this collaboration as document formatting would omit substantial engineering work. Describing AI as an independent production authority would misstate who accepted and operated the result.

The [Author biography](#) is the owner's supplied account. This methodology page is distinct from it. It explains how readers should evaluate the engineering evidence, with the limits of the available record visible. Publication assembly and formatting are a separate activity from building the systems.

Responsibility and the subject of evidence

A useful architecture assigns an owner to consequential state, a consumer to each authorization decision and an observer to each claimed outcome. That assignment is initially a specification. To make an engineering claim, one must identify what executable component actually performs the responsibility, where a failure can occur and how the result was observed.

Three records therefore remain separate. Research records explain a proposition, assumptions and counterexample. Reference records identify an executable model, inputs and observed behavior. Production records identify a dated operational observation and its provenance. The records can support one another, but they are not substitutes. A compiled carrier does not validate all semantic values. A finite checker does not establish the behavior of every product route. A ready workload does not show that a particular customer objective was fulfilled.

The [dated implementation report](#) retains the strongest provenance already published. Its 10 October 2026 examination combined authenticated read-only metadata with retained workload observations and source inspection. That report did not execute a customer action to prove fulfillment. This successor authoring environment does not have product repositories or live infrastructure access; it therefore adds no fresh production observation and does not independently recertify that examination. The owner states that both products

operate end to end for real customers. No payment status, customer identity or independently validated feature deployment is inferred from that statement.

A test can change the claim

The research record contains corrections, failures and withdrawals as well as successful checks. In the recorded September 2026 reference work, boundary tests challenged text normalization, nested structures, container counts and total encoded size. Later passing tests support the corrected bounded codec behavior. They do not establish complete object authorization or a full runtime. The [AD8 investigation](#) keeps those limits explicit.

A particularly instructive failure mode is a test that obtains its supposed observation from the fixture's expected answer. Such a runner can produce a polished success report without executing the behavior under examination. The appropriate engineering response is to withdraw the assurance, retain the scenario as an authored outline and build an independent observation path before claiming execution. A large inventory of scenarios cannot repair that missing path.

Another example is a valid final state reached through an invalid intermediate observation. The [atomic-observation dossier](#) deliberately publishes a partial two-field update. The resulting total is wrong at the first observation and correct at the last. A final-state-only check would pass. This original public example makes the testing lesson inspectable without exposing the product's implementation.

These are concrete reasons for preferring a failed counterexample over an unqualified success label. A failed test should prompt a question about which claim, assumption or implementation must change. The publication does not invent a detailed conversation history about how particular AI proposals were debated. The available evidence supports the corrections and test outcomes; it does not establish every private decision that preceded them.

Working with substantive AI proposals

AI-generated designs and code need the same acceptance boundaries as other candidate work. A proposal can be coherent and still omit an important observation route, broaden a value language, misclassify an uncertain effect or assume progress that an environment does not provide. Agreement among several outputs does not supply the missing experiment.

The research illustrates a disciplined evaluation pattern: state the obligation, expose the assumptions, construct a case that should succeed, construct a case that should fail, inspect the actual implementation and retain the observed result. For AI collaboration, this pattern gives the human integrator a concrete basis for accepting or rejecting a proposal. It is a method supported by the examined corrections and reference artifacts, not a claim that an identical checklist governed every product change.

The new [projection case](#) illustrates one such boundary. Four boxes and four items share a numeral but fail the declared preservation relation. A model or compiler that preserves only the number has not preserved the meaning. A generated program may also compile while

silently narrowing numeric values. Those are questions for explicit validation, not confidence inferred from fluent explanation.

Integration, operations and continuing responsibility

A sole human engineer's responsibility includes the seams between components: deployment configuration, authority boundaries, infrastructure transitions, recovery and acceptance of the integrated behavior. AI contribution can be substantial at each stage while final responsibility remains with the person who directs and accepts the system. This publication claims no staffing equivalent, invented productivity percentage, development duration or autonomous staffing history.

The operating environment connects persistent institutional work to external action. An attempted action, an acknowledged request and a completed obligation have different acceptance conditions. The [uncertain-effect dossier](#) shows why a missing acknowledgement cannot prove that nothing happened. The existing continuity exhibit supplies a more concrete public reference with a controlled witness and restart reconciliation. Neither is substituted for a production customer experiment.

Recovery also does not automatically restore permission. A persistent institution may retain its name and history while its execution incarnation changes. The [epoch case](#) makes that distinction explicit. Likewise, a backup observation does not establish a successful restoration with matching history and continued obligations. These are separate engineering tests to demand before broadening an operational claim.

Credit and novelty

Protection principles, causal ordering, provenance, durable execution, formal methods and proper scoring rules are foundational contributions by other researchers. The [prior-art comparison](#) and [bibliography](#) identify those works and their limits. The contribution claimed here is an authored institutional synthesis and scoped engineering investigations, not invention of every ingredient or proof of superiority.

The attribution to Cuttlefish and Freehold is specific to those systems. Capstone's contributions concern the separate Moguls platform; they are not the engineering team behind Cuttlefish or Freehold. No unrelated project's contribution is repurposed as evidence for these systems.

Acceptance and review

Edition 2026.10.7 extends the preserved predecessor with twelve synthetic investigations. Its public teaching cases are author-executed and externally reproducible; independent reproduction is not claimed. The public evidence schema separates engineering maturity from assurance and rejects missing provenance and unsupported escalation. Build success establishes successful assembly and checks within their scope, not scientific truth or release permission.

The predecessor's issued PDFs, archives, manifests and source companion retain their exact bytes and URLs. New artifacts use successor-specific paths. The correction ledger explains

how current qualifications relate to older records without rewriting them. Publication requires a public-only build, a separately executed private semantic disclosure review and exact owner approval.

For a specialist review, start with [Research Depth & Proof](#), select a claim, run its positive and deliberately faulty case, and compare the observation to the stated nonclaims. Use the [review guide](#) to submit a counterexample or a missing foundational reference. The strongest feedback identifies exactly which inference fails and what evidence would settle it.