LONG-HORIZON AI ENGINEERING UNDER COMPLEXITY

An Inquiry into Principles and a Framework for Keeping Complex Work Coherent Across Time

PREPRINT, COMPILED SEPTEMBER 21, 2026

Franz Bruckhoff

think-bigger.org

ABSTRACT

Until recently, the development of highly complex engineered systems was usually reserved for large organizations. What those organizations supplied was not only labor, but the structure that held their work together across many years. The emergence of powerful agentic AI has redefined the scale of what one person, or a small team of two to three, can realistically attempt and accomplish.

However, complex work over long horizons depends on explicit structure, including bounded components, contracts, verification, and human authority that survive the intervals between decisions and their consequences. AI engineering, throughout this paper, refers to that kind of work where humans and AI agents build complex systems in any domain.

This paper states that structure as five domain-general principles. To ground them, recorded practice was researched for the mechanisms those principles require, drawing on public engineering material from organizations such as NASA, SpaceX, Airbus, Boeing and TSMC, on ISO and IEEE process standards, and on the working practice of leading individual engineers. Of 219 recorded statements, the 118 that passed screening were reduced to 79 generalizable patterns. From those patterns a separately published framework of 106 clauses was written, and every clause names its grounds in a generalized pattern, a principle, or a stated goal.

The paper does not reprint the principles or the clauses. It gives the reasoning behind them, their derivation from recorded practice, and measurements of the framework itself together with a semantic graph built to analyze it.

That graph binds every clause to the exact source bytes it was written from, and its 4,206 nodes and 8,913 edges carry the clauses, their grounds, their terms, and the classified relations between clauses and goals. It exists to analyze the framework statically for internal coherence: whether every clause rests on a stated ground, whether the clauses agree with each other, whether their terms hold together, and whether each requirement serves a defined goal.

Language models produced the graph's logic layer under policies fixed before the corpus run: extraction accepted a result only when two isolated lanes returned an identical output, and recorded a classified gap otherwise, while the clause-to-goal relations were classified in a single reviewed lane and are recorded as such.

Measurement finds every clause grounded and every pattern cited, across 183 grounding links; 25 clauses for which no link to a stated goal was recorded; 39 flagged conflicts between obligations, all adjudicated to a single extraction artifact rather than to contradictions in the text; and four cycles among the clause relations, retained by decision. A rebuild from the frozen inputs returns an identical graph, hash for hash.

The problem addressed in this paper is the coherence of a complex project across years: whether its purpose, its contracts, its authority, and its terms of acceptance still hold when most of the work is carried out by AI agents under human direction. Stronger execution at each step does not produce that coherence; structure that survives between the steps does.

1 INTRODUCTION

1.1 Problem and scope

As of September 2026, long-horizon AI-assisted engineering without adequate support structure and methodology is generally subject to integrity erosion. This erosion can be difficult to detect early on.

While individual tasks may complete correctly, the relationships binding them eventually erode in various ways: They become incorrect, inconsistent, ambiguous, stale, detached or are lost entirely.

For example, a contract might be misread, causing the implementation of an interface against a behavior that was never authorized by the specification. Multiple specifications might state different acceptance criteria targeting the same system behavior with unclear prioritization. Stale requirements might keep interfering long after the underlying decisions have been reversed. Stale tests may keep passing while the criteria to be verified had been changed quietly.

This decay accelerates, and one reason for this is that AI agents operate without institutional memory. While humans absorb unwritten history and implicit consensus into tacit knowledge, AI agents presently only evaluate the bounded context provided to them at execution time.

Each isolated session produces locally optimal work that can remain completely oblivious to the project's accumulated decisions, the grounds and authority behind them, and the cross-cutting commitments made weeks, months or years earlier. As people turn over and tooling evolves, this lack of institutional machine-accessible memory compounds into drift that is effectively irreversible.

To resolve these problems, it is necessary to externalize transient conversational instructions and tacit human knowledge into durable, machine-readable structure, contracts, methods, and records. The key to keeping long-horizon systems coherent across time is to anchor the project's governing invariants and contracts through human authority in a machine-readable way, while automated verification guarantees that no local change can silently compromise the project's global intent.

There are six primary classes of failure that can break the relationships underpinning complex, long-horizon projects: late feedback, design obsolescence, loss of purpose, contract loss, loss of evidence, and loss of authority.

When feedback arrives late, wrong decisions may be repeated before their result becomes visible.

When the implementation changes, knowledge bound to the old design is lost silently while the requirement it served may remain in force without anything left to explain it.

When the purpose behind a requirement is forgotten and nothing in the project can explain why it exists, later work may treat it as arbitrary and delete it as unnecessary.

When a contract between two system components is lost from the context of the work that modifies these components, changing one component can break the other in a way that goes unnoticed by either.

When the evidence for an earlier acceptance is lost, later work may be unable to correctly infer which checks justified that acceptance, and may end up verifying against different criteria than the ones originally used, thereby silently drifting the definition of done without anyone deciding that it should.

When the record of who may have authority to approve a change is lost, an AI executor may approve it without a recorded decision, or the change may be pending indefinitely because it is unclear who holds approval authority.

All of these failures tend to go unnoticed, allowing potentially catastrophic systemic errors to propagate long before anyone detects them.

Each occurrence erodes the ability of the humans and AI agents working on the project to detect a mistake before it is repeated, reconstruct the project's prior decisions, understand why the current design and its requirements exist, anticipate what a change will break, reconstruct what had been verified against what, verify present work, or establish who may authorize the next change. These blind spots allow localized mistakes to compound silently into systemic failure that is hard to resolve.

The six classes of failure discussed in this paper are structural, domain-general, and not unique to software. They generally tend to recur in every domain where complex long-horizon work is being done.

Across all kinds of these domains, the essence of these classes of failures stays constant while the conditions under which they occur may vary.

In the software domain, iteration tends to be cheaper in time and effort than in the hardware domain. Verification is relatively fast, and a change to a software component can usually be reversed more easily than a change to a physical one. The difference across domains is the degree to which iteration is cheap, verification is fast, and changes are reversible. The domain sets the default, while detection speed and reversal cost set the outcome.

To illustrate, imagine a coordination platform for autonomous wildfire-response drones, built and operated over the course of many years by a small team of two, with AI agents carrying out most if not all of the heavy implementation work. In year one, the founders commit to the regional fire authority. Their platform retains decision provenance for five years, such that every decision their platform makes during an incident can be fully reconstructed from raw telemetry.

That commitment requires that raw telemetry is safely stored for five years. The replay tool must deterministically reconstruct any intervention from that telemetry, while any change in telemetry format or operating procedure over time keeps old records replayable. Everything is fine until one day three routine tasks touch those areas. One deletes raw telemetry older than six months and keeps only summaries immediately available. Another task rewrites the replay system to run from those summaries. And finally, a task is executed which migrates the telemetry format but converts only live data, leaving the archived data incompatible. Each task passes its tests, while the commitment has been broken three times because it belongs to no single component. The decision behind it sits in a Decision Record from 2025, and no task in 2029 gains awareness of it. One day, the authority investigates a fatal crash of one of the firefighting drones. As it turns out, it is not possible to reconstruct the platform's decisions from eight months before, because the raw telemetry data was deleted and only summaries were kept.

None of the contracts for telemetry data storage, replay, and format stated the commitment to the regional fire authority: The requirement to retain five years of raw telemetry data, deterministic replay under any circumstance, and compatibility across format and operating procedure changes. No contract required the three changes to keep the commitment, therefore they broke no contract.

The solution is to state the commitment in each of the contracts for telemetry data storage, replay and data format, so that any later change to any of them meets it.

In the separately issued framework, such a commitment is a contract under the principle Specifications as Contracts [1]. A contract here fixes the behavior and acceptance conditions that

humans have approved for other components to rely on [2, Section 2].

But simply restating the commitment in three contracts is not enough. The commitment needs one authoritative source, and the contracts that must uphold it need to point back to that source. Otherwise each contract carries a copy of the commitment, and copies drift.

The commitment to the regional fire authority belongs at the System level. It is one commitment, owned by one named human, and it is the source that the telemetry data storage, replay, and format contracts all serve. Each of those contracts then carries the specific obligation it must preserve. The storage contract carries five-year retention of raw telemetry. The replay contract carries deterministic reconstruction of any retained incident. The format contract carries continued replayability of old records across format and procedure changes. Each of those contracts points back to the same top-level commitment, so none of them can be satisfied locally while the commitment is broken globally.

Under Verification-First Engineering, the commitment must be checkable before any of the three changes begin. An independent verifier takes a past incident, replays it from stored raw telemetry, and records whether the reconstruction is deterministic and complete. After a format or operating procedure change, the same check must show that old records still replay. If the check fails, the change is not done. The six-month deletion cannot pass as a storage detail, because the check is against the recorded commitment, not against the local tests of the storage component. The same holds for the replay rewrite and the format migration.

This is what the framework for keeping complex work coherent across time requires of every commitment: a durable record of the purpose it serves, and relations that let a proposed change be traced to everything it can affect.

In the fire-authority example, the storage, replay, and format contracts would each point back to the commitment and the verification that checks it. A proposed change would point forward to every contract, check, and record it can affect.

A task that deletes raw telemetry data after six months would point forward to the storage contract, the fire-authority commitment, and the replay check that depends on the retained data. The AI agent would see that the deletion weakens a commitment it is not authorized to change, and the change would stop before it is accepted. The same would happen for the replay rewrite and the format migration. None of the three could pass on its own tests alone, because its own tests were never written against the commitment it was about to break.

This is the difference between a project that stays coherent over a long horizon and one that drifts without anyone deciding to let it. The commitment to the fire authority is a single commitment, not three. It has one named human owner, and it is reachable from every contract that must uphold it.

Three checks apply to the three changes, and all three evaluate the work against the Acceptance Criteria that express the fire-authority commitment. The check that gates changes to the storage system fails before the change that would expire raw telemetry data is accepted. The check that gates changes to the replay system fails before the change that would rewrite the replay system to run from summaries is accepted. The check that gates changes to the telemetry format fails before the change that would convert only live telemetry data is accepted.

The checks stop the changes because they run against the recorded commitment, not against the component's own tests.

Human authority applies across all five principles. In the example, a change may compress telemetry, change how it is stored, or migrate its format within the recorded commitment. Only the responsible human may shorten the five-year retention period or approve replay from summaries, because those changes weaken a commitment made to the fire authority [2, Sections 1.3 and 15.2]. Passing the tests of a single task does not authorize a change to a commitment those tests never checked.

Two requirements are utilized to examine these commitments.

The first is *standing grounds*: each requirement has a durable record of the purpose or prior commitment it serves, with relations that can be followed in both directions. For the platform, the retention requirement would carry the fire-authority commitment it serves, and that commitment would name every requirement that rests on it.

The second is *traversable consequence*: recorded relations let a check find the commitments, records, and tests that need review before a proposed change. For the six-month deletion, those relations would lead from telemetry storage to the fire-authority commitment, its replay check, and the format contract that relies on it.

The five principles organize the work; these two requirements concern the grounds of its commitments and the consequences of changing them.

The same coordination question arises when a physical system changes: which requirements must still hold after a part is replaced, and what evidence will establish that they hold? The form of the evidence differs; the requirement that work is incomplete until verified does not. A software component test and a physical inspection produce different Verification Artifacts. Rules shared across domains must preserve that difference while keeping purpose, interfaces, verification, and authority explicit. The framework places those relations in five domain-general principles, and domain practice supplies the methods used to check a particular system [1].

1.2 Questions and contributions

Existing engineering practice provides mechanisms for keeping these commitments available: component boundaries, specifications, verification records, and controlled changes. The mechanisms examined in this paper come from public engineering material of organizations that build complex systems, ISO and IEEE process standards, and the working practice of leading individual engineers.

A generalized pattern states a mechanism without tying it to one organization or tool.

In their source organizations, the mechanisms are carried by review bodies, process groups, and people who remember why a

rule exists. An individual or small team and multiple AI agents have no such carrier around them. For that individual or team, the mechanisms must survive as written rules and records that humans and agents can follow.

The separately issued Specification turns the generalized patterns and the five principles into requirements for project records and their relations [2]. It separates what a record must contain from the file or format that holds it. Its first complete domain binding is a software reference layout.

Two questions follow.

How can mechanisms already used in practice become engineering rules that apply across domains while giving a project specific requirements to follow?

What do the recorded grounds and relations of those rules reveal about missing support, conflicting requirements, and the effects of a proposed change?

The measurements in this paper examine the issued documents. Whether their rules keep other projects coherent remains a separate empirical question.

The Specification must itself retain the grounds and change relations it requires of projects [3]. A machine-readable copy of its requirements, linked to their source text, records those relations for inspection.

The inspection shows where grounds are recorded, where relations to goals are missing, and which requirements need review together. Section 7 reports the findings with the evidence and review decisions that support them.

Three contributions follow.

- **Formulation.** Five principles in a separately issued framework, with reasons for each rule, limits on its use, and a division of human and AI authority across all five.

- **Artifact.** A separately issued Specification derived from recorded practice, with clause-level grounds and a machine representation linked to the source text.

- **Measurement.** Released checks and findings that let a reader inspect the recorded grounds, locate gaps and conflicts, trace the recorded effects of a change, and repeat the recorded checks from fixed inputs.

WORK IN PROGRESS

The complete paper is pending release.