Applying Systems Engineering Discipline to Generative AI

A controlled workflow for technical analysis, fidelity control, and engineering continuity

James E. Dalton II, CSEP, CSSBB

ORCID 0000-0002-8564-124X

Abstract

Generative AI can provide substantial analytical leverage in complex engineering work, but its usefulness is constrained by predictable failure modes including unsupported inference, context drift, loss of precision, and eventual conversational limits. This paper presents a systems-engineering-based methodology for using generative AI as a fallible analytical component within an engineer-controlled multi-thread architecture. The approach combines explicit problem framing, bounded analytical execution, interactive challenge review, human adjudication, external configuration control, rationale preservation, fidelity monitoring, drift correction, and deliberate continuity and rehydration mechanisms. Thread separation is functional and contextual rather than autonomous; the human engineer remains the integrating element across the architecture and retains technical authority throughout the process. Accepted engineering state and decision rationale are maintained in controlled external artifacts rather than entrusted to conversational memory. The objective is to engineer the surrounding workflow so that substantial analytical capability can be used without surrendering engineering judgment or accepting unearned trust.

Keywords: Generative AI; Systems Engineering; Technical Analysis; Configuration Management; Risk Management; Engineering Continuity

1. Introduction

Generative AI can be an extraordinarily powerful engineering tool. It can analyze large bodies of information, expose relationships, challenge assumptions, compare alternatives, synthesize technical material, and perform substantial analytical work at a speed that would otherwise be difficult to achieve.

It is also fallible. It can misunderstand a constraint, introduce an unsupported assumption, lose precision as a conversation grows, produce a highly plausible statement that is not actually supported, or gradually drift from an established technical baseline. Long-running conversations also have practical limits. Eventually, a deeply developed working thread will need to be replaced.

For serious technical work, those realities cannot simply be ignored.

My approach has therefore been to apply systems-engineering discipline to the AI-assisted workflow itself: identify known and predictable failure modes, build controls around them, continuously monitor fidelity, preserve authoritative engineering state outside the conversation, and maintain a deliberate continuity mechanism when conversational limits are reached.

The objective is not to make the AI authoritative or autonomous. The objective is to engineer a workflow in which its capabilities can be used aggressively while the human engineer remains actively engaged as the integrating element, exercises engineering judgment throughout the process, and retains technical authority over what ultimately becomes accepted state.

2. Start by Engineering the Problem

For difficult technical work, a prompt is not necessarily a question. It may function much more like a task specification.

The quality of the work depends heavily on the quality of the problem definition provided to the model. Objectives, constraints, assumptions, terminology, authoritative source material, known facts, unresolved questions, boundaries, and expected outputs should be made as explicit as practical before significant analysis begins. Sometimes several paragraphs are sufficient. For complex work, the initialization material may extend for many pages.

The point is not length for its own sake. The point is to reduce ambiguity.

This also means distinguishing carefully between different kinds of information. What does an authoritative source actually establish? What is an engineering inference? What is an assumption? What remains uncertain? What is merely a hypothesis worth investigating?

A generative model can produce convincing language around all of those categories. The engineer must prevent the language from blurring the distinctions between them.

One of the simplest and most important questions throughout the process is: **What supports this statement?**

There is a meaningful difference between *the source establishes this*, *this conclusion follows from the source*, and *this appears plausible*. Where traceability matters, that distinction should remain visible. When information is insufficient, uncertainty should be exposed rather than silently filled with a plausible answer.

That discipline begins before the first analysis and continues throughout the work.

3. Fidelity Monitoring and Drift Control

Monitoring fidelity is not a final step in this workflow. It is a governing discipline.

Every substantive AI response is read. Constraints, terminology, reasoning, and conclusions are continually compared against the established problem and authoritative information.

Even small inaccuracies matter. An incorrect name, a subtly altered constraint, an imprecise characterization, or an unjustified increase in certainty may be harmless in isolation. It may also be an early indication that the working context is beginning to drift.

I therefore treat small errors as potential leading indicators rather than harmless conversational noise. Correct them early.

If inaccuracies begin accumulating, simply correcting the latest sentence may no longer be enough. The appropriate response may be to reintroduce authoritative material and deliberately re-ground the conversation in the known engineering state.

This is the same basic reasoning applied elsewhere in engineering: do not wait for a small deviation to become a system-level failure before responding to it.

4. Separate Execution, Challenge, and Decision Authority

For substantial work, I use an engineer-controlled multi-thread architecture rather than an autonomous multi-agent system. Separate conversation threads are assigned deliberately different functions, but their separation is functional and contextual, not autonomous.

The primary conversation serves as the principal integration workspace. It holds the evolving problem context, supports discussion, and is where new analytical tasks are framed before being dispatched.

When a bounded piece of analysis is needed, the assignment is developed in that primary conversation. The resulting task definition, together with whatever source material is required, is then transferred deliberately to a separate worker thread.

No thread is assumed to know what occurred elsewhere. Context transfer between threads is intentional. Each working context receives the source material, constraints, prior conclusions, and instructions necessary for the specific role it is being asked to perform.

A bounded task does not necessarily mean a small task. It means that the objective, scope, source information, constraints, and analytical purpose are controlled.

A complex system can therefore be examined through multiple deliberate analytical lenses. One pass may examine dependencies. Another may look specifically for failure modes. Another may search for contradictions, evaluate alternatives, assess risk, identify single points of failure, or test whether a proposed simplification destroys an important relationship.

A worker thread may perform substantial analysis with relatively little intervention once the assignment has been adequately framed. Its output may itself become a significant technical artifact. Its size does not change its role: it remains an analytical input, not an authoritative engineering decision.

The worker output then moves to a separate review thread with a deliberately different mandate: **attack the work**.

Look for unsupported assumptions. Find contradictions. Identify missing constraints. Challenge weak reasoning. Find claims that sound plausible but are not actually substantiated. Identify important alternatives that were not considered. Determine where the analysis may have exceeded the available evidence.

The review process is not necessarily passive. The engineer may question a criticism, ask what evidence supports it, challenge an assumption made by the reviewer, introduce a constraint that changes the assessment, or require the reviewer to reconsider a finding. The purpose of the thread is not simply to generate objections; it is to create a separated context in which the work can be actively challenged and interrogated.

I do not describe this as an independent review in the strict sense. The reviewer may be operating on the same underlying model family and may therefore share systematic limitations with the worker. The value comes from separation of conversational context and analytical purpose. The review thread is not being asked to extend or defend the original work. It is being asked to challenge it.

Human involvement is not confined to initial task definition or final approval. The engineer remains actively engaged across the architecture—defining tasks, transferring context, monitoring outputs, questioning results, challenging critiques, supplying missing information, refining rationale, adjudicating findings, and controlling what becomes authoritative engineering state.

The threads provide separated working contexts. The human engineer provides integration, judgment, and authority across them.

Review findings ultimately return to the primary conversation, where adjudication occurs. The reviewer is not automatically correct because it found a criticism. The worker is not automatically correct because it produced the original analysis. Findings are discussed, compared against authoritative information, accepted where valid, rejected where unsupported, and refined where necessary.

Only after that process is complete does accepted material become part of the controlled technical record.

The core technical loop is therefore:

Primary conversation → bounded worker thread → separate review thread → primary conversation → human adjudication → controlled technical documentation

The human engineer remains engaged throughout that loop and retains technical authority at every stage.

5. The Conversation Is Not the Authoritative Repository

A long technical conversation naturally contains exploration. It contains ideas that were considered and rejected. It contains mistakes that were later corrected. It contains preliminary analyses, superseded language, unresolved questions, and reasoning that may have been useful on the way to a conclusion without becoming part of the final result.

That makes the conversational thread extremely useful as a working environment.

It also makes it a poor configuration-controlled repository.

Accepted engineering state is therefore captured outside the conversation in controlled documents.

When useful, final accepted text can be requested in fenced code blocks or similarly portable formats so that it can be transferred cleanly into the appropriate external artifact without carrying conversational material along with it.

The important distinction is simple:

The conversation contains working state. The controlled documents contain authoritative state.

But authoritative state is more than the final technical answer. To preserve engineering continuity, it must also contain enough of the reasoning behind important decisions to make the accepted state intelligible later.

6. Preserve the Rationale, Not Just the Answer

Complex engineering decisions frequently make sense only when the reasoning behind them is preserved.

A later reader—or a newly initialized AI conversation—may understand that a particular decision was made without understanding why it was made, which constraints drove it, what alternatives were examined, what earlier assumptions were corrected, or why an apparently reasonable alternative had already been rejected.

That creates an obvious continuity risk.

For that reason, I use a separate rationale thread after significant technical decisions have been adjudicated and the accepted technical state has been captured.

Relevant developments are transferred deliberately into that thread. Sometimes the transfer can be accomplished directly by describing what was just worked through and providing the relevant technical output. After a long analytical session, however, the primary conversation can create a structured handoff describing what was examined, what changed, what was decided, which constraints mattered, and what material the rationale thread needs in order to reconstruct the decision faithfully.

The objective is fidelity, not ceremony.

The rationale thread is not simply given a decision and asked to document it autonomously. The engineer works interactively with the thread to reconstruct and refine the reasoning. The developing rationale may be questioned, corrected, expanded, or redirected until it accurately captures why decisions were made, which alternatives mattered, what was rejected, what corrections occurred, what remains unresolved, and what engineering intent must survive into the continuity record.

That distinction is important. Preserving a conclusion without preserving the reasoning that gives it meaning may retain configuration while losing engineering intent.

Once the rationale accurately represents the decision history, it is again reviewed by the human before being placed into controlled rationale or continuity documentation.

The supporting continuity loop is therefore:

Human-adjudicated technical state → controlled technical documentation → deliberate context transfer → interactive rationale development → controlled rationale documentation → primary conversation

The technical and rationale artifacts serve different functions, but they ultimately become part of the same controlled continuity record.

Engineering continuity requires more than knowing where the work is. It requires enough preserved reasoning to understand how it got there.

7. Continuity and Controlled Rehydration

Long-running AI-assisted work creates a predictable lifecycle condition: eventually the active conversation will reach its practical limit.

That should not become an emergency.

Continuity is engineered while the work is still active rather than reconstructed after the thread becomes unusable.

I maintain a Continuity folder containing the controlled artifacts needed to recover the engineering state in a clean conversation. Those artifacts are maintained in sequence because sequence carries information.

A technical baseline may establish what was known at one point. A later rationale document may explain why that baseline changed. A subsequent artifact may record a correction, new analysis, revised doctrine, or an unresolved issue that altered later decisions.

Reordering those documents only by subject can destroy part of the reasoning path.

Some artifacts establish accepted technical state. Others preserve decision rationale. Others may preserve governing doctrine, distinctions, or rules by which subsequent work is supposed to be evaluated. Together, they preserve both the developing engineering state and the meaningful path by which that state was reached.

The same controlled artifacts can also be used before a full thread transition becomes necessary. If an active conversation begins accumulating assumptions or losing fidelity, authoritative material can be reintroduced to re-ground it in the established baseline.

When a thread finally reaches the end of its useful life, the continuity package becomes the basis for deliberate rehydration.

The first artifact is **Document Zero: Rehydration**.

Document Zero does not describe the engineering problem itself. It defines how the recovery process will operate. The new thread is told that recovery documents will be provided sequentially, that each document must be ingested and checked before proceeding, and that substantive technical work is not to begin until the rehydration process has been completed.

Document One: Initial Context / Problem Definition then establishes the original problem. Depending on the work, this may include the objective, constraints, assumptions, terminology, authoritative sources, boundaries, expected outputs, governing rules, and other information necessary to understand the problem as it was initially framed.

Subsequent documents restore the evolving technical state and reasoning architecture. These may include accepted technical baselines, analytical findings, doctrine, decision rationale, corrections, unresolved issues, checkpoints, and other artifacts developed during the life of the work.

They are provided in their controlled order.

The purpose is not simply to tell the new thread what the final answer is. It is to reconstruct enough of the problem's meaningful evolution that the new working context understands both the present state and the reasoning framework by which that state should be interpreted.

Document transfer is verified as it occurs.

Uploading an artifact is not the same thing as verifying that its usable content is actually available to the model. Asking, “Did you read everything?” provides little assurance.

Instead, ingestion is checked through observable evidence. Depending on the artifact, that can include identifying the document structure, confirming material from the beginning and end, sampling distributed interior sections, identifying major headings or tables, describing distinctive content from different regions, and explicitly reporting truncation, extraction failures, unreadable sections, or other gaps.

The human compares those results against the actual source.

This does not mathematically prove that every token of the source exists in the working context, nor should it be described that way. It provides observable evidence from which the engineer can judge whether the transfer appears sufficiently complete and faithful for the work being performed.

If there is doubt, change the transfer mechanism. Split the document. Reload it in smaller sections. Verify again.

Once one artifact has been satisfactorily ingested, the AI asks whether additional recovery documents remain. The process continues until the human declares rehydration complete.

Only then does substantive technical work resume.

This methodology has been exercised across multiple successive thread transitions during long-running, deeply developed work. As those efforts progressed, the continuity package grew with them.

In one later recovery cycle, the accumulated continuity record extended to thousands of pages and required more than twenty ordered recovery artifacts. Because those artifacts had been developed and controlled throughout the work rather than reconstructed after failure, a clean thread could be rehydrated sequentially and the ingestion of each artifact checked.

In that recovery, productive work resumed in under an hour with essentially no meaningful loss of established working context.
The important point is not the number of pages or documents.
It is that conversational continuity did not have to be trusted as the system of record.
The engineering state—and enough of the engineering intent behind it—had been deliberately made recoverable.

8. Engineering the Workflow Around Fallibility

None of this makes a generative model infallible.
That is not the objective.
The model can still make mistakes. A worker can produce weak analysis. A reviewer can raise an invalid criticism. A long conversation can drift. A document transfer can fail. A thread will eventually reach its practical limit.
The methodology assumes those things can happen.
The engineering response is to:

- Control their consequences.
- Frame the problem carefully.
- Monitor fidelity and control drift.
- Transfer context deliberately.
- Separate execution from challenge.
- Remain actively engaged within each working context.
- Retain human technical authority.
- Capture accepted engineering state outside the conversation.
- Preserve not only decisions but the rationale that gives them meaning.
- Prepare continuity before it becomes necessary.
- Verify recovery rather than assuming it.
- Then resume.

The workflow is recognizably systems-engineering in structure.
Problem framing resembles requirements and problem definition.
Bounded analytical passes resemble decomposition and analysis.
Separated working contexts provide functional specialization while human integration maintains technical coherence.
Challenge reviews provide a verification function.
Human adjudication performs technical decision-making.
External baselines provide configuration control.
Rationale capture preserves engineering intent and decision traceability.
Fidelity monitoring and re-grounding provide anomaly detection and corrective control.
Continuity and controlled rehydration provide deliberate state recovery.
Risk management spans the entire process: predictable AI failure modes are identified in advance, controls are designed around them, and the workflow is structured to reduce the likelihood, propagation, consequence, and recovery cost of failure.
The terminology may differ because the working medium is generative AI, but the underlying engineering disciplines are familiar.
Generative AI can provide extraordinary analytical leverage when used this way. But the capability does not come from delegating engineering authority to the model or making the model increasingly autonomous. It comes from engineering the surrounding process so that useful capability does not require unearned trust.