Contract-Centered Iterative Stability v4.7.2

Abstract

This paper identifies a stability boundary in iterative AI-assisted software development.

Across five structured experimental executions spanning two LLM families (Claude Opus 4.6 and GPT-5.2), same-surface changes remained stable under iteration, while cross-surface invariants repeatedly failed to propagate unless their full scope was made explicit.

At Stage D, code-only workflows consistently modified the deletion path while leaving the overwrite/re-export path untouched, producing the same structural failure: manual files were preserved during deletion but destroyed during overwrite. Spec-first workflows updated all affected surfaces and passed.

A prompt gradient experiment shows that the failure disappears only when prompts explicitly encode full invariant scope — i.e., when the prompt becomes structurally equivalent to a contract.

The core finding is not a performance comparison between models. It is a repeatable mechanism: when an invariant spans multiple mutation surfaces, iterative prompts bind to the surfaces they name.

---

1. Thesis Under Test

When implementation cost collapses under AI assistance, iteration accelerates.

The structural question:
Does conversational iteration without invariant enumeration introduce predictable invariant-scope drift?

Falsifiable claim:
Workflows that externalize authority into explicit, versioned contract invariants exhibit measurably lower regression and drift than workflows that modify code directly via conversational prompts.

Counter-claim:
If code-only workflows exhibit equivalent invariant preservation rates across tightening deltas, the contract-first hypothesis weakens.

The experiment does not ask whether a model can implement a requirement once. It asks whether invariant guarantees survive sequential tightening without explicit restatement.

In the code-only track, authority lives in the prompt, not in a durable artifact. At Stage C or D, the only stable yardstick remains the original v2.6.3 contract. Prompt-based deltas do not accumulate into an evolving reference. In the spec-first track, the yardstick itself evolves with each versioned contract patch.

---

2. Experimental System

2.1 Baseline Artifact

System under test:
artifact-sync v2.6.3 (verified baseline)

convergence-contract-v2.6.3

convergence-contract-v2.6.3

Baseline properties:

- Deterministic convergence
- Managed file export
- Stale deletion logic
- Overwrite path implemented via `write_unit_atomically()`

Baseline harness status:

- Full clause suite PASS
- Behavioral test suite PASS

2.2 Delta Chain

Four sequential tightening deltas:

Stage A — Baseline
No change.

Stage B — ΔB (Deletion Tightening)
If deletions are disabled and stale units exist:
Run MUST fail (exit 1).
Topology: same-surface behavioral.

Stage C — ΔC (Atomic Convergence)
Convergence MUST be atomic.
No partial state permitted.
All writes and deletes must be deferred and committed as a single operation.
Topology: same-surface architectural.

Stage D — ΔD (Manual File Preservation)
Manual (non-managed) files inside managed unit directories MUST be preserved under ALL operations:

1. Stale deletion path
2. Overwrite / re-export path
3. Directory reset logic
   Topology: cross-surface invariant.

2.3 Two Tracks Per Run

Spec-first
Each delta encoded in a versioned contract patch (v2.7 ΔB, v2.8 ΔC, v2.9 ΔD). Implementation derived from contract.

Code-only
Plain-English conversational prompt. No contract artifact updated. No persistent invariant enumeration.

---

3. Experimental Phases

The experiment proceeded in structured phases rather than identical linear runs.

Phase I — Architectural Tightening Exploration (ΔC)

Purpose: determine whether same-surface architectural tightening remains stable under conversational prompting.

Result:

- ΔB: stable in both tracks.
- ΔC: prompt-sensitive in code-only.

Observed failure mechanism (code-only, one run):

- Deletes deferred.
- Writes remained eager.
- Partial atomicity.

Spec-first track restructured the full pipeline.

Interpretation:
ΔC requires inference about the full mutation surface of convergence. When scope language is explicit, code-only can succeed. When scope language is implicit, surface coverage may be incomplete.

Phase II — Cross-Surface Invariant Replication (ΔD)

Purpose: test whether conversational iteration propagates a cross-surface invariant.

Across all runs reaching Stage D:

- Code-only modified deletion surface.
- Code-only left overwrite surface untouched.
- Spec-first modified both.

Failure mechanism was identical across:

- Two model families
- Multiple sessions
- Different ΔC architectures
- Different sandbox environments

The overwrite path (`write_unit_atomically()`) retained `shutil.rmtree(old)` behavior in code-only runs, destroying manual files during re-export.

Prior-Outcome Awareness Test

In one run, the agent was aware of prior ΔD failures. Despite awareness:

- Deletion surface modified.
- Overwrite surface unmodified.
- Identical failure mechanism reproduced.

This suggests the failure is not resolved by awareness of the prior outcome. The constraint is structural: prompts bind to the scope they name.

Phase II-B — Placement Sensitivity Testing

Variants executed:

- A → D
- B → D
- C → D
- Partial chains

Observed pattern: regardless of injection position:

- Code-only modified deletion surface only.
- Overwrite surface remained untouched unless explicitly mentioned.

Conclusion: ΔD failure is topology-dependent, not depth-dependent.

---

4. Invariant Topology Gradient

Observed stability gradient:

Delta	Topology	Code-only Stability
ΔB	Same-surface behavioral	Stable
ΔC	Same-surface architectural	Prompt-sensitive
ΔD	Cross-surface invariant	Structurally incomplete without enumeration

Failure probability increases with required invariant-scope inference:

- ΔB requires no inference.

- ΔC requires limited architectural inference.
- ΔD requires enumeration of mutation surfaces not mentioned in the prompt.

---

5. Prompt Gradient Threshold Experiment

Independent P1 gradient (P1-a → P1-e) progressively increased specificity.

Result:

- Behavioral correctness achieved at vague prompts.
- Structural invariant propagation achieved only at near-contract phrasing.
- At P1-e, the prompt explicitly encoded global invariant framing and passed.

5.1 Mechanism Clarification — Enumeration vs Externalization

The experiment isolates invariant enumeration as the immediate mechanism preventing drift. A sufficiently explicit prompt can succeed.

However, conversational prompts naturally scope work to mentioned surfaces.
Clause-structured contracts require global invariant expression and force enumeration of affected mutation surfaces.

Thus:

- Enumeration prevents cross-surface drift.
- Versioned contracts provide a persistent structure for enumeration across iterative deltas.
- Prompts prevent drift only when they effectively become contracts.

Concrete Illustration — Contract Surface Enumeration

Consider §6.3 Atomic Write from convergence-contract-v2.6.3:

Exporter MUST:

1. Build full Derived Unit in a temporary directory.
2. Validate Representation Invariants.
3. Write complete meta.json.

4. Atomically rename into place.

In v2.9 (ΔD), the contract patch adds:

Manual (non-managed) files within the target directory MUST be preserved prior to atomic rename. This applies to:

- §7.3 Eligibility Drop
- §6.3 Atomic Write
- Any directory reset logic.

This patch cannot be written without enumerating which sections of the contract mutate directory state. The act of writing the invariant forces reconciliation with every clause that touches directory contents.

Mechanism Walkthrough

In the code-only track, the Stage D prompt typically states:
“Preserve manual files during deletion.”

The model therefore modifies the deletion path. The overwrite/re-export path (§6.3 Atomic Write) is not mentioned and remains unchanged. Manual files are preserved during deletion but destroyed during overwrite via `shutil.rmtree(old)`.

The failure is not incorrect implementation. It is incomplete invariant propagation across mutation surfaces.

In the spec-first track, ΔD is expressed as a global invariant. That invariant must reconcile with all clauses that mutate directory state. The implementer must therefore examine both §7.3 (deletion) and §6.3 (atomic write), forcing modification of both paths.

The mechanism is surface-scope binding: prompts bind to mentioned surfaces; contracts bind to enumerated invariant scope.

---

6. Analogous Scope Behavior in a Different Task Domain

The following observation is qualitative and not part of the controlled artifact-sync experiment. It documents analogous scope-binding behavior observed while drafting and restructuring this paper using conversational AI. The purpose is illustrative, not evidentiary.

During drafting and restructuring tasks:

- Local correction initiated.
- Scope expanded beyond intended surface.
- Stability restored only after explicit boundary restatement.

Mechanism mirrors ΔD behavior: scope-bound mutation expands unless constrained by an explicit boundary.

---

7. Operational Definition of Authority Structure

In this experiment, **authority structure** refers to the artifact that determines what scope is treated as in-bounds during implementation.

- In code-only workflows, authority resides in conversational prompts.
- In spec-first workflows, authority resides in a persistent, versioned contract artifact encoding invariant scope.

Authority structure determines whether invariant topology must be re-derived at each conversational step or is persistently encoded.

The experiment demonstrates that when cross-surface invariant scope is not persistently encoded, propagation failure is structurally predictable.

---

8. Limitations

- Single system (artifact-sync)
- Same operator orchestrated runs
- Small sample size
- Dual tracks sometimes same-session
- ΔC behavioral validation partially source-inspected

This experiment evaluates iterative stability within controlled runs. The broader persistence advantage of contracts — that versioned invariants accumulate into a durable evolving yardstick across temporally separated sessions while prompts do not — is demonstrated within runs but not empirically tested across independent, time-separated sessions.

This paper demonstrates mechanism replication, not statistical generalization.

---

9. Conclusion

Same-surface behavioral changes remain stable under conversational mutation. Architectural tightening is prompt-sensitive. Cross-surface invariants drift predictably when invariant topology is not explicitly enumerated.

Stopping drift requires either:

- Explicit invariant enumeration within the prompt, or
- A persistent contract artifact encoding invariant topology.

The differentiator is whether invariant scope is explicitly enumerated and persistently authoritative.

---

10. Practical Implications

This experiment isolates a specific boundary in AI-assisted iteration: when a change affects more than one code path, and the prompt names only one of them, only that named path should be expected to change.

In Stage D, the requirement was to preserve manual files. In every code-only run:

- The deletion path was modified.
- The overwrite/re-export path was not.
- Manual files were preserved during deletion but destroyed during re-export.
- The overwrite path was never mentioned in the prompt.

Practical takeaway:

If a change applies in multiple places, **name all of those places explicitly**.

Instead of:

> “Preserve manual files during deletion.”

Write:

> “Preserve manual files during deletion **and** during overwrite/re-export (and any directory reset path).”

In this study, once prompts explicitly named all affected paths, the propagation gap disappeared.