Models should verify each other

Key takeaways

1. The builder should not supply the final confidence judgment on its change.
2. Each review round records the change; only the builder may write.
3. Bound the review rounds: converge with evidence or escalate to a human.

Common misunderstandings of cross-model verification

Which claim would you defend right now? Choose before the evidence.

"Sharing the builder's diagnosis helps the reviewer check the fix faster."

"Read-only repository access makes a recorded change snapshot unnecessary."

"The review should keep going until the reviewer approves the change."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

Your agent finished the change — who says it is correct?

The builder wrote the code and the story; it should not also supply the verdict

The change is done — and the builder has certified its own work

```
You: "Cache entries leak between tenants. Fix the cache key."
Agent: edits src/cache.py, adds a test, runs the suite.
       All tests pass. "The fix is correct."
```

Why self-certification is weak	What an independent reviewer changes
The builder’s diagnosis anchors its re-check	starts fresh from the artifact
A model-family blind spot re-applies	a different model challenges it
Helpful tone drifts toward approval	the contract asks for findings, not praise

A strong default: the builder does not supply the final confidence judgment.

Dotfiles provides the review skill and interface

A skill maps a natural-language request to a tested workflow and executable

IN CODEX, SAY

```
"Do a Claude review."
```

IN CLAUDE CODE, SAY

```
"Do a Codex review."
```

- Skill prompt files · ~/dotfiles/codex/skills/claude-review/SKILL.md and ~/dotfiles/claude/skills/codex-review/SKILL.md interpret the natural-language request.
- Executable scripts · ~/dotfiles/bin/{claude-review,codex-review} run the protocol; ~/dotfiles/bin/{claude-ask,codex-ask} launch the peer reviewer.

Instapoll

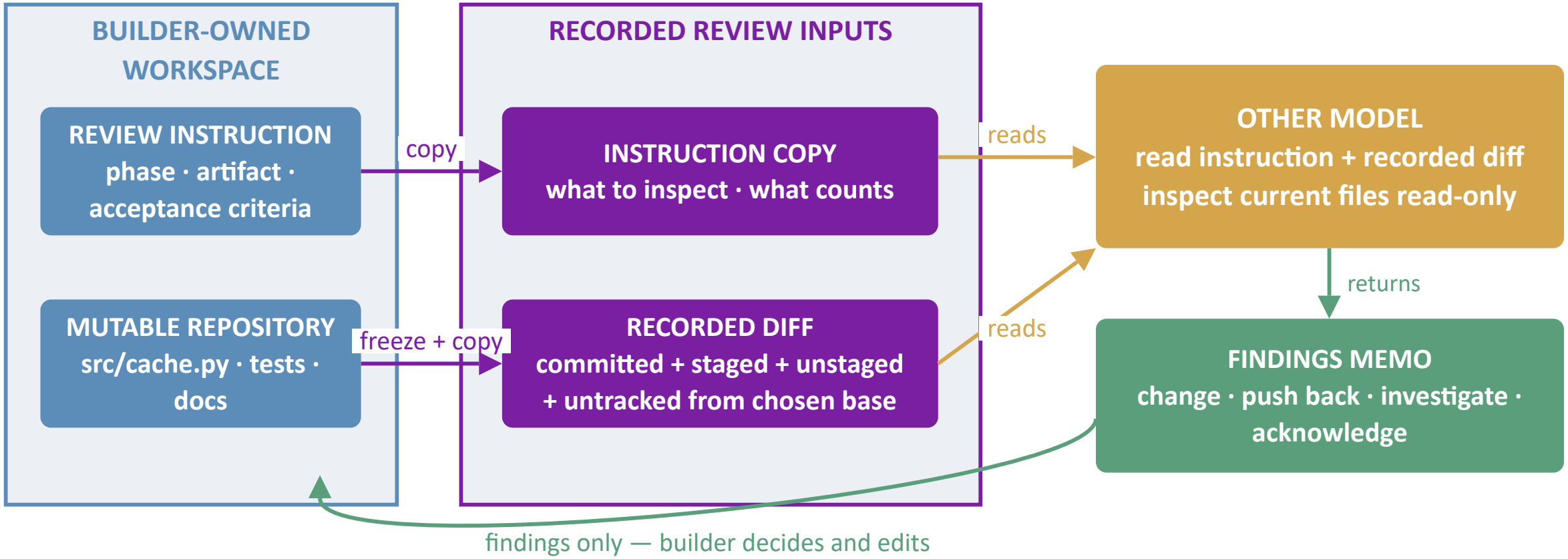
Instapoll · Review · Sep 1 B

Codex implemented a change. Which review setup gives you the most independent challenge before you accept it?

- A. Start a fresh, read-only Claude review given only the change and the code as it stands
- B. Ask Codex in the same conversation to confirm that its change is correct
- C. Give Claude Codex’s full transcript and ask whether it agrees with the diagnosis
- D. Give the reviewer write access so it can silently repair anything it dislikes

Each round records the change; only the builder may write

Recorded diff for provenance; read-only repository access for context



A review returns structured feedback

The dotfiles review SKILL imposes the format: every finding has a class and severity

CLASS · WHAT KIND OF PROBLEM?

```
PREMISE
  requested goal or assumption is wrong

QUALITY
  artifact contains a concrete defect

TOOLING
  problem is outside the builder's worktree
```

SEVERITY · WHAT RESPONSE IS REQUIRED?

```
BLOCKER
  work cannot be accepted until resolved

IMPROVE
  answer with a change, investigation, or
evidence

NIT
  minor issue; acknowledgment is enough
```

Class identifies the problem; severity says what the builder must do.

The builder answers every finding exactly once

Four response headings make missing or repeated finding numbers easy to detect

REVIEWER REPORT · FIXED FIELDS

```
#1 QUALITY/BLOCKER · src/cache.py:42-43
failure: cross-tenant user_id aliases entries
required: tenant_id in key + collision test

#2 QUALITY/IMPROVE · tests/test_cache.py:18-27
required: cover two tenants, one user_id

#3 QUALITY/NIT · docs/cache.md:1-14
required: clarify key format
```

BUILDER RESPONSE · EACH NUMBER ONCE

```
## Addressing
  #1 -> tenant_id in key + collision test
## Pushing back
  #2 -> test_tenant_isolation:18 discriminates
## Investigating
  None
## Acknowledged
  #3 -> clarify docs
```

Missing, duplicated, or invented finding numbers make the memo invalid.

The builder can push back for good reason

The reviewer verifies the cited reason in the next round

REVIEWER · FINDING #2 (IMPROVE)

```
Location: tests/test_cache.py:18-27
Claim: no test covers two tenants
  sharing one user_id
Required action: add a collision test
```

BUILDER · PUSHING BACK

```
## Pushing back
- #2 -> test_tenant_isolation:18
  creates user 7 under tenants A, B
  and asserts distinct entries.
  It fails against the old key, so
  it discriminates. Coverage exists.
```

If a blocker survives the round cap, stop and hand the exact disagreement to a human.

Disagreement is resolved by evidence, not argumentation

Ask: “Explain the disagreement in plain language. Define any technical terms you must use.”

Reviewer claim	Builder response	Resolution
Bug at a cited line	reproduce the failure	fix + regression test, or refute
Coverage is missing	add the case or cite the existing test	reviewer verifies next round
The premise itself is unsafe	stop; surface assumption + consequences	a human decides, and the decision is recorded
Same blocker at the cap	state the exact unresolved claim	escalate — the cap does not certify

The useful product is an auditable evidence chain: claim → location → disposition → evidence → human acceptance.

source: OpenAI, Introducing upgrades to Codex; Separating signal from noise in coding evaluations

Key Takeaways

1. Ask for an independent read-only review; each round records the change.
2. Disposition every finding: fix it, push back with evidence, or escalate.
3. Bounded rounds converge or hand you the disagreement — approval stays human.