AI Deliberation & VS Code

Review before there is a diff to defend.

Called from the agent already in your workspace. Frontier models from different labs argue — you get the answer and where they split.

A WORKFLOW GUIDE

QUORUM.DOG

What is in here

This paper assumes you already have a review culture — pull requests, CI, owners, people who will read your diff. Sections 03 to 05 are the argument for what that culture cannot reach. 06 and 07 are what you check in and what you say. 08 to 10 are the three jobs. 11 to 13 are the artefact and the wiring. If you are short of time, read 04 and 06.

EXHIBITS

03

What Quorum is, in one page.

Quorum is a deliberation platform. One question goes to several frontier models from different labs at once. They answer independently, critique each other, and are judged — and what comes back includes **where they disagreed**.

If you have only ever used one assistant, the distinction that matters is this: asking the same model a second time gives you the same reasoning again, more politely. Independent models fail in different places. That is the whole mechanism, and it is the same reason your team asks two people rather than one person twice.

A SEAT

One frontier model on the panel, reasoning independently. Three seats is the standard panel and the convention used throughout this paper; the response is a list, not a fixed trio. Quorum names seats as shapes rather than vendors.

A MODE

The configuration of a deliberation — which engines sit, how many rounds, how deep. Not a knob on a single model: a repeatable decision setup you select, and one your team can standardise on.

A RECEIPT

The record of a completed deliberation: which model sat in each seat, judge scores, difficulty, cost, latency. It is the part that attaches to a pull request. §11.

WHERE IT RUNS

Quorum's own surfaces, an OpenAI-compatible REST API, and a hosted MCP server. VS Code speaks remote MCP natively, which is the whole of the integration. §12.

THE CLAIM, STATED NARROWLY

Quorum does not claim a single model is bad. It claims that a single model, however capable, **cannot be its own second opinion** — it has one way of being wrong, and it applies that way consistently across every file it touches in your repository. Panels exist for the questions where that matters.

AND WHERE IT DOES NOT APPLY

Most of what you ask an agent is execution, not judgement. Quorum is deliberately wrong for renames, formatting, import order and syntax — and its own `estimate` call will say so before you spend anything, returning `likely_hurts` (§13).

04

// before the diff exists

You already have review. This is the read it cannot get.

Your team disagrees for a living. There is a pull request template, a CI pipeline, an owners file and at least one person who reads diffs properly. This paper does not propose replacing any of it.

But notice what review is. It happens **after** the approach is chosen; by the time a diff exists, arguing with the approach means asking someone to throw work away, so mostly nobody does. It is bounded by who is available and who still has context. And it anchors on your framing — you wrote the description, and it tells the reviewer what to check.

A panel is none of those things. It reads the question you hand it, three ways, with no shared history to anchor on and nothing built yet to defend.

One point worth being precise about: **the agent in your editor is not one of the panel.** It stays where it is — holding the workspace and your checked-in instruction — and calls in independent models, none of which is the one you were just talking to. The agent has your repository; Quorum has whatever paragraph it is handed. Turning one into the other is the agent's real job.

CHAT · AGENT MODE SRC/API/TOKENS.TS

We need a tenant id on SessionToken. Draft the change.

```
read src/api/tokens.ts  ·  41 call sites in this repo
this changes an interface other code depends on — convening the panel

quorum · deliberate                                    28.5 s
question: "Adding a required tenantId to SessionToken.
  41 call sites we own; a mobile client and one partner
  integration we do not. The release cannot be rolled
  back. Optional first and tighten later, or required now?"
```

Three seats. Two split on **sequencing** — optional-then-required against required-now. The third argues the field belongs on the request, not on the token. Nothing has been written yet; tell me which way to go.

Exhibit 01 — Where the panel enters a turn the agent is already handling. The agent stays in your workspace and calls models it is not one of. Invented turn, shown to illustrate the shape.

BEFORE THE DIFF

The strongest objection arrives while the approach is still **cheap to change**.

NO SHARED CONTEXT

Three reads with **nothing anchoring them** to how you framed it.

THE ABSENT REVIEWER

For code whose owner left — or whose owner **is you**.

AGREEMENT COUNTS

Independent models landing together is **information too**.

05

The five moments it escalates on.

None of these is a language or a framework. They recur in any codebase with more than one author and a history longer than anyone's memory of it.

01	**A module with no living owner**	The author left, the owners file has no entry for that path, and the tests pass. Whoever reviews your change will be reading the module for the first time in your pull request. The panel is the read nobody on the team is in a position to give. **§08 works this one end to end.**
02	**A change that binds every caller**	A schema, a public signature, an auth boundary, a wire format — anything with a migration behind it. Reviewers exist, but each one checks the slice they own and approves it honestly. Being confidently wrong here is expensive, and independently trained models genuinely disagree about it. **§09.**
03	**A fix that may be treating a symptom**	The agent proposes a change, the suite goes green, and the diff is three lines. Whether those three lines address the cause is a question a diff has no room to ask and a reviewer has no evidence to answer. **§10.**
04	**An approach you want attacked, not rewritten**	You already have a design and you are fairly sure of it. Hand the panel the proposal and ask what breaks. The thing you are buying is whether three models find the *same* weak spot — not another version of the design.
05	**A review thread that has stalled**	Two engineers, forty comments, both arguing well. A panel is not a casting vote and should never be used as one. It is three independent reads on the technical question, which is how you find out whether the thread is about the technical question at all.

THE COMMON SHAPE

Every one is a moment where the work is understood and the **judgement** is what remains. The three jobs in §08 to §10 run in one direction: what review cannot reach. No reviewer with context, then every reviewer with partial context, then a question no comment box has room for. **Unowned, shared, structural.**

AND WHAT IT SHOULD NEVER ESCALATE

Renames, formatting, import order, syntax, test scaffolding, anything with one right answer. Your agent is faster and better at those and a panel adds a minute for nothing. Quorum's own classifier agrees: it returns `likely_hurts` when the answer is a verbatim artifact a synthesis pass would only rewrite — which is most of what a code assistant is asked to do.

06

Teach it once — and check it in.

Give the agent a standing rule for when to convene the panel and the decision stops being yours to remember. This is the whole setup.

Put it wherever your workspace keeps persistent agent guidance — a checked-in instructions file if your team has one, otherwise at the top of the session. The wording is the same either way; what changes is who else gets it.

STANDING INSTRUCTION — PASTE AS WRITTEN, THEN EDIT THE LAST LINE

```
You have a Quorum connector. It convenes a panel of frontier models
from different labs and reports where they disagree.

Use it on judgement, not on work you can do yourself.

Convene the panel when:
  - a change touches something other code depends on: a schema, a
    public signature, an auth boundary, a wire format;
  - I am working in a module no one on the team currently owns;
  - you have proposed a fix and are not certain whether it
    addresses the cause or the symptom;
  - I ask you to attack an approach rather than implement it.

When you do:
  - put the real code in the question - the signature, the failing
    assertion, the migration - not a description of it;
  - state the constraint that is not in the diff: the callers we do
    not control, the release we cannot roll back;
  - report where the seats disagreed and do not reconcile it;
  - give me the strongest argument against the way I am leaning.

Do not convene it for renames, formatting, import order, syntax or
test scaffolding. Do those yourself.

Escalate without asking me first.   ← or "ask me first"
```

WHY THIS ONE GETS COMMITTED

In a personal editor a standing instruction is a preference. In a shared repository it is **policy, and it goes through review like everything else**. Committed, the whole team escalates on the same conditions, a new joiner inherits them on clone, and when somebody thinks the bar is wrong they argue about it in a pull request with a reason attached.

THE LINE THAT MATTERS MOST

“Put the real code in the question.” The panel argues about what it is given. An agent left to summarise will write *a change to a shared interface*; you want *a required tenantId on SessionToken, 41 call sites here, a mobile client and a partner integration elsewhere, no rollback on this release*. Same escalation, an entirely different argument comes back.

07

Steering it in the moment.

The checked-in rule handles the moments nobody spotted. These are for the moments you did — when you already know the decision is worth an argument and you want one before you open the branch.

"Before I open the PR — put this approach to the panel and give me the strongest argument against it."

Forces an attack, not a review

"Do not rewrite it. Attack it. Do all three seats find the same weak spot?"

You want the overlap, not a redesign

"You have given me a fix. Get a second read on whether it is the cause or the symptom."

Asks the question a green suite cannot

"This thread has forty comments. Ask the panel what the disagreement is actually about."

Not a casting vote — a diagnosis

"Send the real signature and the call sites, not a summary of them."

When precision is the whole question

"Where did the seats disagree? Do not reconcile it — put the split in the PR description."

The follow-up worth making a habit

BEST PRACTICE

Use both. The checked-in instruction is the safety net; these are the times you reach for it deliberately. If you only ever do one, commit the instruction — the escalations you would have typed by hand are the ones you already knew were hard.

WHAT MAKES AN ESCALATION LAND

Paste the code, not a paraphrase of it. Say what the change binds and what you cannot roll back. Say which way you are leaning — a panel given a position to attack argues far harder than one handed an open question. And ask for the split explicitly; a tidy synthesis is the least useful thing a panel can return, and the one it will produce if you do not ask.

08

UNOWNED → SHARED → STRUCTURAL

The module nobody owns.

You have been asked to add a second currency to the billing reconciler. It is nine hundred lines, it has not changed in over a year, and the person who wrote it does not work here.

WHAT THE REPOSITORY WILL TELL YOU

```
$ git log -3 --date=relative \
    --format='%ad  %an  %s' -- src/billing/reconcile.ts
2 days ago      renovate[bot]  chore: bump provider sdk
19 months ago   a.morrow       fix: partial refunds
22 months ago   a.morrow       feat: reconcile ledger

$ git shortlog -sn -- src/billing/reconcile.ts
    38  a.morrow  <- no longer at the company
     4  renovate[bot]

$ grep billing .github/CODEOWNERS
(no match)
```

INVENTED REPOSITORY STATE · ILLUSTRATIVE

WHAT THE AGENT SAYS

"The module reconciles the internal ledger against provider settlement events, retries on gaps, and has 42 passing tests. Adding a currency looks straightforward."

All true, and none of it is the question. The agent can read every line. What it cannot do is disagree with itself about which lines are load-bearing — and there is nobody on the team who can do it for you either.

WHAT THE PANEL RETURNS

▲ **The retry loop is what carries the module.** It assumes provider events arrive in settlement order. That assumption is written down nowhere and everything downstream depends on it — a second currency with a different settlement delay breaks it.

■ **Disagrees about where the risk sits.** The code is defensive throughout. The fragile part is the suite: it asserts against a fixture recorded when the module was written, so it will keep passing after you change the behaviour it was meant to protect.

● **Neither is the binding constraint.** Whatever you change, the diff will be approved by somebody meeting this module for the first time. Write the ordering assumption down before you write the change, so review has something to argue with.

Two seats argued about which line is dangerous. The third said the danger is that nobody can review it — and that is the one that changes what you do next.

ELAPSED

A panel takes a median of **28.5 seconds**, 88.2 at the ninetieth percentile. You spend it once, before the branch exists.

P50 28.5 S · P90 88.2 S · N=3,845 · 2026-08-20

Exhibit 02 — A module with no living owner, in three commands, and what three seats said about it. Seat positions are illustrative of the pattern, not a transcript of a specific call.

09

UNOWNED → **SHARED** → STRUCTURAL

The change that binds every caller.

A required field on a shared type. The compiler proves every caller in this repository is correct, and says nothing about the two that are not in it.

THE DIFF

```
--- a/src/api/tokens.ts
+++ b/src/api/tokens.ts
@@ -14,4 +14,5 @@ export interface SessionToken {
   id: string
   subject: string
   scopes: Scope[]
+  tenantId: string
 }
@@ -22,2 +23,3 @@ export function issue(
   scopes: Scope[],
+  tenantId: string,
 ): SessionToken
```

41 call sites here. One mobile client and one partner integration elsewhere. This release does not roll back.

INVENTED DIFF · ILLUSTRATIVE

THE REVIEW IT GOT

@PLATFORM — APPROVED

"Required is right. Every call site in the repo passes it — the compiler would have caught anything that did not."

@DATA-ENG — APPROVED

"No impact on us. We read scopes and ignore the rest of the token."

@MOBILE — REVIEW NOT REQUESTED

—

Nobody here is wrong. Both reviewers answered the question they were asked, correctly, about their own slice.

WHERE THE PANEL SPLIT

▲ **Required is a breaking change whatever the type system says.** The compiler protects the callers in this repository and nothing else. Ship it optional, backfill the outside clients, tighten in a second release.

■ **Disagrees on sequencing.** Optional-then-required is two migrations, and the second one is the one that never gets scheduled. Make it required now, while there is appetite, and hold it to the release train.

● **Both are arguing about rollout.** The prior question is whether the field belongs on this type at all: a tenant is a property of the request, not the token, so every future consumer inherits a field it has to ignore.

Review had already approved this twice — correctly, and on a narrower question. The seats split on sequencing and the third refused the framing entirely.

DO THIS BEFORE THE BRANCH

Worth most when there is no diff yet. If the branch already exists, run it anyway and put the split in the description, so review starts from the argument.

Exhibit 03 — An interface change and the review that approved it twice. Diff, thread and seats invented.

10

UNOWNED → SHARED → **STRUCTURAL**

Patch, or symptom.

An intermittent failure, a three-line fix, a suite that goes green. Green is the weakest evidence on this page.

TERMINAL — BEFORE AND AFTER THE PROPOSED FIX

```
$ npm test -- src/billing

  PASS  reconcile > matches ledger to events      12 ms
  FAIL  reconcile > handles out-of-order settle  1,204 ms
    AssertionError: expected 2 ledger entries, got 3
      at src/billing/reconcile.test.ts:88:5

  Tests   1 failed | 41 passed (42)

— agent proposes: dedupe by providerEventId before write —

$ npm test -- src/billing

  Tests   42 passed (42)
```

INVENTED TEST OUTPUT · ILLUSTRATIVE

WHAT THE DIFF WILL LOOK LIKE

Three added lines, a green pipeline and a reviewer who has thirty seconds. There is nothing in the pull request for anyone to object to.

Which is the problem. **"Should this diff exist at all?" is not a line comment.** It is a question about the design the diff sits inside, and review has no slot for it.

WHAT THE PANEL RETURNS

▲ **Symptom.** Deduplicating at write time makes this ledger correct and leaves the event pipeline delivering duplicates. The next consumer of that stream inherits the same bug, and will not have a failing test to find it with.

■ **Disagrees — the patch is the fix.** Deduplicating at the boundary is the correct design for an at-least-once stream; the defect is that this consumer ever assumed exactly-once. It belongs in the shared helper, not in this file.

● **Neither can be settled from what is here.** One test failed once and passes now. Nothing shown establishes that the third entry is a duplicate rather than a genuine second settlement. Reproduce it first; both answers are premature.

Two seats split on where the fix belongs. The third refused the question as posed — and reproducing the failure is the only one of the three answers that is cheap.

THE PROGRESSION, CLOSED

No reviewer with context (§08). Every reviewer with partial context (§09). A question no comment box has room for (§10). Each one is a place where a working review process is structurally unable to produce the objection you need.

Exhibit 04 — A suite that goes green, and what that does not prove. Test output and seat positions are invented to illustrate the pattern.

11

The receipt, and where it goes.

Every deliberation leaves a record. On a team that is the difference between “the AI said” and something the next person to open this file can read.

WHAT GET_RECEIPT RETURNS

```
request_id          rq_8f2c41a9
mode                quorum-standard
difficulty          0.78

seat 1              judge score, model
seat 2              judge score, model
seat 3              judge score, model

cost                billed, per call
latency             31.2 s
```

SHAPE OF THE RESPONSE · ILLUSTRATIVE VALUES

The judge scores are the part people underuse. They tell you whether the seats **actually disagreed** or converged immediately — which is how you learn that a question did not need a panel, and how you tune the bar in your checked-in instruction rather than arguing about it from memory.

Quorum names seats by shape rather than by vendor on anything shareable, so a receipt can go into a public repository, a customer-facing changelog or an audit file without carrying somebody else’s trademark.

IT EXPORTS

From Quorum, a completed deliberation renders as a full transcript PDF, a carousel, cards or a short video — the seats, their positions and the split, laid out.

IT TRAVELS

The transcript is what goes in the pull request description, the decision record, or the ticket. The argument travels with the change instead of evaporating in a chat pane nobody else can open.

IT ANSWERS THE QUESTION LATER

“Why is tenantId on the token?” is a question someone asks in eighteen months. A receipt is the only artefact in this workflow that is still there to answer it.

> WHERE THIS IS GOING, STATED AS DIRECTION AND NOT AS A FEATURE
>
> A receipt records one deliberation. What it does not do is accumulate: the decision, the disagreement, the option taken, and eventually whether it held. A team that kept that would be able to ask what its own past judgements were worth — which is the question a code review process never gets to answer either. Quorum does not do this today. There is no store, no outcome linkage and no retrieval, and it is named here as the direction the receipt makes possible, not as something you can switch on.

Exhibit 05 — A receipt.

12 Connect it.

Quorum is a hosted MCP server: no package to install, nothing running on your machine. Which is why the config below belongs in the repository.

A workspace file gives the whole team the same server on clone and is reviewable like any other config. The same block works in a user profile if you would rather keep it personal.

.VSCODE/MCP.JSON · WORKSPACE — OR YOUR USER PROFILE

```
{
  "servers": {
    "quorum": {
      "type": "http",
      "url": "https://www.quorum.dog/mcp",
      "headers": { "Authorization": "Bearer ${input:quorum-key}" }
    }
  }
}
```

CHECK THE TOP-LEVEL KEY BEFORE ANYTHING ELSE

VS Code expects `servers`. Claude Code and Cursor nest the identical block under `mcpServers`. Copy a config from a teammate and it is not rejected with a useful message — the server simply never appears in the tool list.

WHY ${INPUT:} AND NOT THE TOKEN

VS Code prompts for the key on first use rather than storing it in a file you might commit. You can hardcode it, but a checked-in API key is a bad afternoon — and this file is going into the repository on purpose.

WHAT APPEARS IN THE TOOL LIST

TOOL	COST	WHAT IT DOES
`estimate`	Free	Classifies and prices a question without running it. §13.
`list_modes`	Free	Which panels your key may convene, and what they cost.
`deliberate`	Paid	Convenes the panel. The one that costs, and the one you wait for.
`get_receipt`	Free	For a finished call: per-seat model and judge score, difficulty, cost, latency.
`run_test`	—	Mode testing. On the hosted server since 2026-08-22.
`certify_mode`	—	Mode certification. Neither is part of day-to-day work.

ONE CALL PROVES THE WHOLE WIRING

Ask the agent to call `list_modes`. It is free, needs a valid key, and returns the panels you are entitled to — so one real answer proves URL, key and wrapper together.

TWO THINGS THAT LOOK LIKE FAULTS AND ARE NOT

A **GET** returns 405 — Streamable HTTP, POST only, and correct. The connection is stateless, so there is no session to keep warm. The same key works at `quorum.dog/v1`.

13

When the pipeline does the asking.

Everything so far assumes a person at the keyboard. For one engineer on one branch that is right: when the decision matters, price is not what you weigh.

A pipeline changes the arithmetic. A check on every pull request touching a shared path convenes the panel hundreds of times a week, with no one there to judge.

That is what **estimate** is for — free, about 0.9 seconds. Its `deliberation_value` is a better gate than price.

It reads whether arguing helps this question at all. The threshold stays yours.

WHAT ESTIMATE HANDS BACK

FIELD	WHAT YOU BRANCH ON
`deliberation_value`	`likely_helps` / `likely_hurts` / `unknown`. The gate. `likely_hurts` fires when the answer is a verbatim artifact a synthesis pass would only rewrite.
`basis`	Returns `"hypothesis"` with an evidence line: a per-task pattern observed at n=36. It is **not a measured effect** — a prior you may override, not a result.
`difficulty_score`	0 to 1. Set your threshold from your own distribution, then move it with evidence.
`estimated_price_usd`	What the call would cost, so a budget ceiling can sit in front of it. `billed_usd` is always 0.

.GITHUB/WORKFLOWS/SECOND-OPINION.YML · YOUR PIPELINE, CALLING TWO OF ITS TOOLS

```
- name: second opinion on interface changes
  if: contains(steps.changed.outputs.files, 'src/api/')
  run: |
    v=$(quorum estimate "$Q" | jq -r .deliberation_value)
    case "$v" in
      likely_helps) quorum deliberate "$Q" | gh pr comment -F - ;;
      likely_hurts|unknown) echo "skipped" ;;   # your call
    esac
```

Exhibit 06 — A gate you write. Quorum ships no CI integration. Invented step; the wrapper command is not real.

FAIL OPEN, ALWAYS

If the classification call fails, let the pull request through. Do not error and do not take the expensive path by default.

TUNE IT WITH RECEIPTS

If receipts on escalated pull requests keep showing no dissent, narrow the path filter: you are paying for agreement.

Commit the standing instruction to one repository you already care about. You will know within a week which of the five moments is yours.

WHERE TO GO NEXT

The escalation router guide at `quorum.dog/docs` covers this branch.