ARCHITECTURE CASE STUDY 7B / 01

AI orchestration and independent quality assurance

A controlled architecture connecting human approval, AI assistants, development tools and independent verification.

Context and objective

The main challenge in AI-assisted software delivery is not code generation alone; it is clear responsibility for decisions, edits and acceptance. Uncontrolled agent handoffs can lead to competing writes, unsupported success claims and inconsistent test environments.

The target was a traceable workflow with a human approval authority, one implementation agent where useful and a separate verification step.

Solution architecture

Project owner: defines objectives and grants approvals. ChatGPT: architecture, coordination and independent acceptance. Desktop Commander: authorized bridge to the Windows machine. Claude Code: optional implementer within an isolated project. Poracode/Crossagents: subagent handoff tested only as a proof of concept.

Instructions, execution, evidence and review were separated. Project isolation and publishing gates formed explicit design constraints.

COMPONENT AND RESPONSIBILITY FLOW

Request + approval	Design + delegation	Isolated implementation	Independent QA

Design principle

Separate decision rights, execution and acceptance. The solution is intentionally bounded: protect existing systems and make authority auditable rather than assume unlimited autonomy without evidence.

Implementation and integration

Selected technical steps, interfaces and design choices without exposing protected implementation details.

Implementation approach

An intentionally defective even-number filter was used in an isolated Node.js project. A single Claude thread repaired a narrowly scoped function and passed four tests after the change.

Exactly one Claude reviewer was spawned through Crossagents. Its findings returned to the originating thread automatically. This was observed behavior, not proof of a second model provider or enforced read-only access.

Quality assurance and acceptance gates

A separate deterministic validation designed by ChatGPT checked the repaired function using 1,000 generated arrays, negative values, zero cases and a mutation check. This provided independent assessment without claiming a second provider operating inside Crossagents.

The evidence chain covers failure and recovery, the 4/4 original test outcome, review handback and the additional verification suite.

Architectural rationale

The solution supports incremental change and small, observable delivery steps. Each gate has a concrete acceptance condition. Failures and limitations remain visible instead of being concealed behind a green status label.

Verification strategy

- Record a baseline and limit the change scope.
- Check the integration path actually in use, not just an isolated component.
- Exercise negative cases and explicitly forbidden states.
- Preserve findings, residual uncertainty and reproducible handover records.

Evidence, risk and learning

A credible reference separates actual observations, isolated tests and operational objectives that remain open.

Security boundaries and remaining limitations

One writer per change, isolated project folders, no production project reconfiguration and explicit cost and permission boundaries. The Crossagents reviewer was reported as having “full-access”; read-only behavior was prompt-level guidance only.

Neither a cross-provider autonomous review inside Crossagents nor automatic resumption of an existing ChatGPT conversation was demonstrated.

Impact and personal contribution

The case illustrates a verifiable orchestration pattern: define roles and authority, reproduce failure, perform a minimal repair, run independent checks and document remaining limits.

My contribution was solution design, authorization boundaries, implementation coordination and acceptance of AI-assisted work, not a claim to have hand-coded an entire multi-agent platform.

Transferability

The pattern can inform AI integration, internal automation and developer-experience tooling: explicit interfaces, least privilege, documented state, cost control and reproducible verification. Applying it elsewhere requires system-specific security and acceptance work.

Lessons learned

Meaningful architecture evidence describes more than an end state. It explains how defects were discovered, what a test can and cannot establish, and who has authority over production release. Documentation therefore forms part of the engineering deliverable rather than a marketing afterthought.

Evidence basis: internal project records dated 22–25 November 2026, test reports and portfolio register. This application edition intentionally excludes private paths, tokens, repository URLs and personal system information.