C01 Trust Calibration & Mental Model

Conceptual Beginner

How to think about your relationship with Claude to get the most out of it

The right mental model

The most productive way to think about Claude Code: a very fast junior developer with encyclopedic knowledge of patterns, but without the judgment that comes from project experience. It can produce 100 lines of solid code in 10 seconds, and make a bad architectural decision in the same breath.

This framing has an immediate implication: supervision is not optional. It is calibrated according to the type of task.

Calibration by task type

Task	Trust level	Action
Renaming, formatting	High	Accept without reviewing
Test generation	Medium-high	Check coverage
Module refactoring	Medium	Review the diff
Architecture decision	Low	Mandatory validation
Critical business logic	Low	Systematic review

The general rule: the more irreversible or costly to fix a decision is, the more validation is needed.

The unsupervised iteration trap

The cost of an error grows quasi-exponentially with the number of unsupervised iterations. A bad data structure choice at step 1 will contaminate all 8 subsequent steps.

The recommended practice: validate in small increments. After each significant change, look at the diff before continuing.

```
# Check what Claude actually modified
git diff --stat
git diff src/
```

You are the “main thread”

Claude Code executes. You orchestrate. The distinction is not stylistic, it is structural: you define the tasks, priorities, and validation criteria. Claude produces the artifacts. This division is what enables parallelizing multiple Claude instances on independent tasks.

What you always keep: architecture decisions, library choices, acceptance criteria definition.

What you delegate: implementation within a defined scope, boilerplate generation, mechanical refactorings.

Rewind: undoing a bad direction

If Claude heads in a problematic direction, `/rewind` lets you return to the previous state before re-iterating. Better to rewind early than to build 10 more exchanges on a shaky foundation.

```
/rewind    # Undo the latest changes
           # Return to the last healthy point
```

Pattern Amplification

Claude reproduces the patterns it finds. In a well-structured codebase, it produces coherent and idiomatic code. In a disorganized codebase, it amplifies the disorder. If your code lacks clear patterns, provide them explicitly in `CLAUDE.md` rather than hoping Claude will infer them.