Treat Code as History

Why AI-driven code editing falls apart, and how ai-desk fixes it.
Explained with pictures, not jargon.

ai-desk — Hiroyuki Okinoi · 2026-05

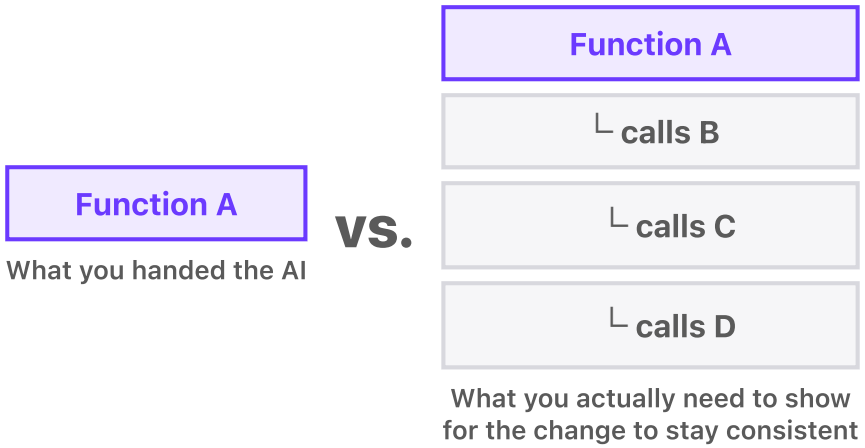
1 Sound familiar?

- You ask the AI to "fix this function" — **something else breaks**
- After 3 rounds of edits, **the structure has fallen apart**
- Ask "understand the whole codebase" — it **hallucinates**
- The AI "DRYs" things up — and you **can't track what changed**

It's not that the AI is bad. The way you hand it code is wrong.

2 Why it breaks — in one line

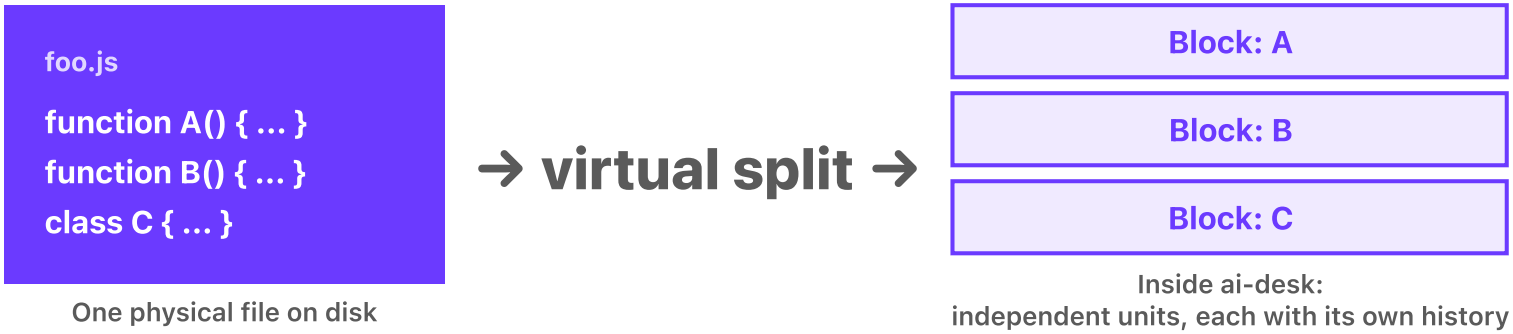
Because the AI never sees everything related at once.



Fix A alone, and B/C/D stay stale → mismatched edges → broken code.

3 One file, virtually broken into pieces

The file on disk stays **as one file**. ai-desk **splits it in its head** into chunks (Blocks).

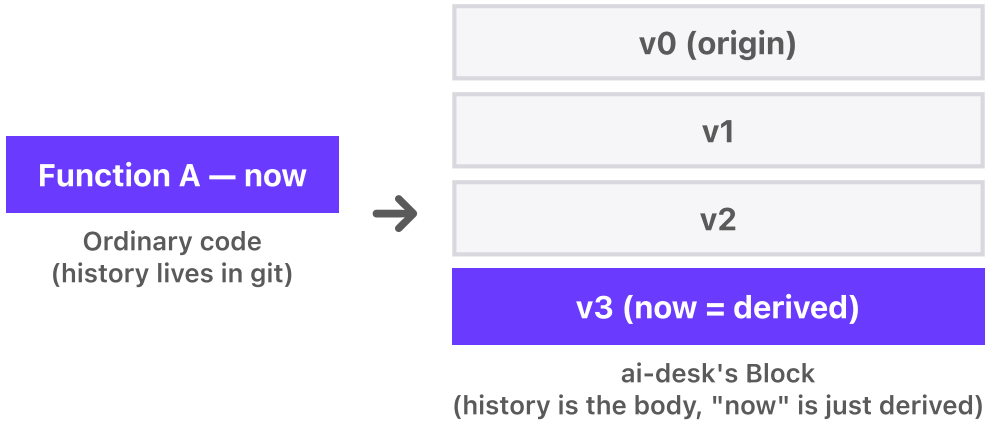


The file **is never split for real**. The AI's workbench is fine-grained.
When you write back, it **recomposes into one file**. Best of both worlds.

4 Shift 1 — Code is "history", not "current state"

Normally: a file = the current state.

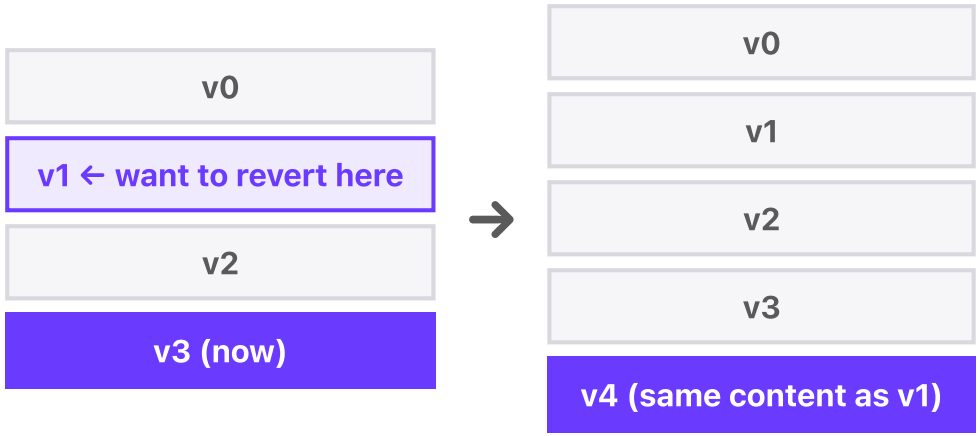
ai-desk: **a chunk (Block) = the entire ordered list of past versions**.



"The current code" is just **the last page** of the history. The body is the whole stack.

5 Because it's history — going back doesn't mean deleting

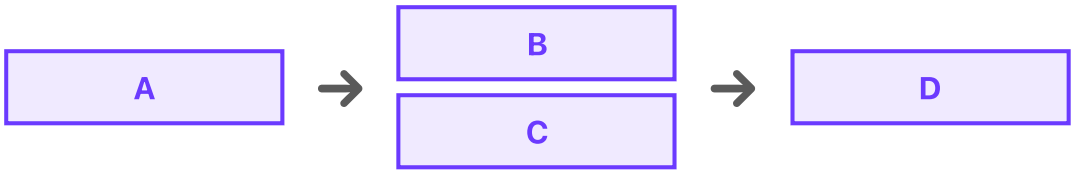
To go back, you don't **erase** the past — you **copy that page forward** as a new entry.



You can iterate fearlessly. The past is never destroyed,
so "undo" becomes **a new step forward** — additive, not destructive.

6 Shift 2 — Make relationships explicit too

Which Block calls which? ai-desk holds this as a **map you can walk without reading the code**.



- **"If I change A, what breaks?"** → walk the map forward, instant answer
- **"Where did this bug come from?"** → walk it backward
- **"What did the map look like at time T?"** → combine with history, restore

7 Killer feature — bundle the dependencies for the AI

When you say "fix A", ai-desk hands the AI A together with B, C, D — everything A depends on, in one prompt.

The AI sees **callers and callees together** — so it can keep both sides consistent.

8 And then redistribute automatically — this is the heart

The single edited blob coming back is **mechanically split** and appended to each Block's history.

One AI edit updates every related Block at once.
The classic "fixed A but B got left stale and broke" failure is **structurally impossible**.

9 Three pieces, locking together

1

Chunks (Blocks)

Functions, classes, rules — all the same kind of unit. The AI reads **one shape**.

2

History (Versions)

Each Block keeps its own past. **You can always go back**, so big edits become low-risk.

3

Map (Refs)

Relationships are explicit. You can **bundle "everything related"** and hand it over in one shot.

When these three click together,

the result is: **large AI edits stop collapsing**.

10 Tags & priorities — the AI narrows what it has to read

Every Block carries **tags** (e.g. `#high` `#logic` `#io`) and a **priority**.

The AI **searches** and reads only the slice it actually needs.

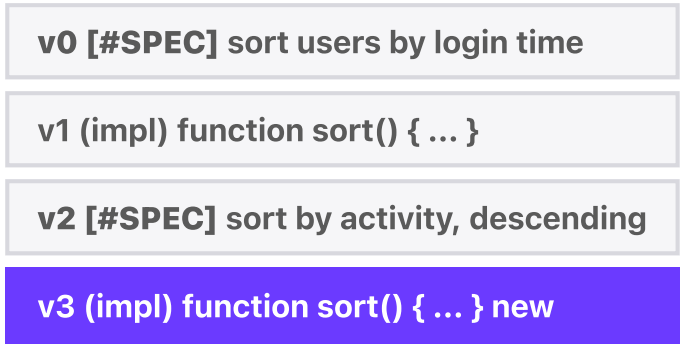
Even on a huge codebase, the AI's field of view is **narrowed**.

You no longer need to invoke "read everything and tell me" — the hallucination mode.

11 The SPEC tag — the spec lives inside the Block

A version tagged `#SPEC` contains **only the spec**, no code.

The initial spec and every spec change stay in the Block's own history.



> "What was this function **supposed to do**?" → just read the Block's history.
> **Spec, implementation, and the trail of changes** all live in one place. External docs going stale stops being a problem.

12 Not a replacement for git — it sits on top

GIT'S JOB

- File-level history
- For humans collaborating
- Branches, merges, PRs

AI-DESK'S JOB

- **Function-level** history and map
- For AI editing
- Bundling "everything related" into one prompt

Use both. git is the **archive**. ai-desk is the **workbench the AI touches**.

13 It works — this isn't a manifesto

105
tests passing
(end-to-end)

4
single-HTML demos
(card game / spreadsheet / node graph / kanban)

0
external deps
(Node standard library only)

All four demos were **written by the AI in one session**.

That's itself the proof that "no collapse" is real.

History + Map + Bundle

With these three pieces, AI coding escapes
"fix one function, break the surroundings".

No new language. No build step. No dependencies.
Just **change the way you hand things to the AI**.

github.com/AoyamaRito/ai-desk

Hiroyuki Okinoi · pen name Aoyama Rito