コードを「歴史」として持つ

AI にコードを書かせる時に、なぜ崩れるのか。
ai-desk の解き方を、絵だけで理解する。

ai-desk — 沖井広行 · 2026-05

1 こんな経験、ありませんか?

- AI に「この関数直して」と頼んだら、**別の場所が壊れた**
- 3 回修正してもらったら、**構造が崩壊した**
- 大きいコードベースで「全体把握して」と頼むと、**幻覚を吐いた**
- AI が DRY 化したあと、**何が変わったか追えなくなった**

これは AI が悪いんじゃなく、渡し方が悪い。

2 なぜ壊れるか — 一言で

AI には関係する全部を一度に見せていないから。

A だけ直しても B・C・D が古いまま → 噛み合わなくて壊れる。

3 １ファイルを「仮想的に」分解する

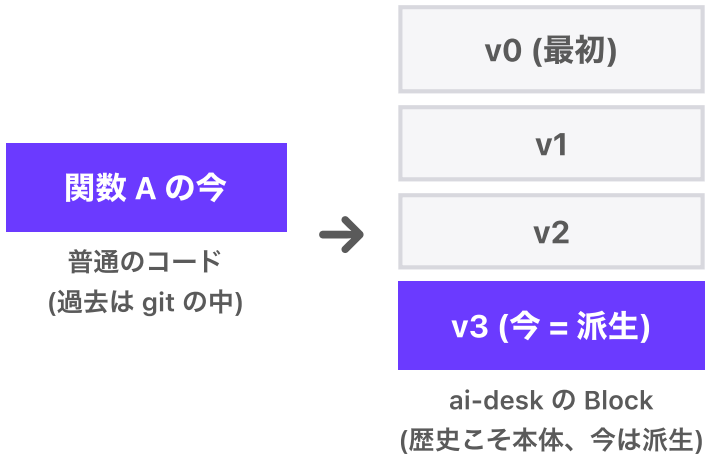
ファイルは 分けない。ai-desk が 頭の中で かたまり (Block) に切り分ける。

物理ファイルは **触らない**。AI が触る作業台では細かく分かれている。
編集が終わったら **１ファイルに戻せる**。両方の良いとこ取り。

4 発想の転換 ① — コードは「今の姿」でなく「歴史」

普通: ファイル = 今の状態。

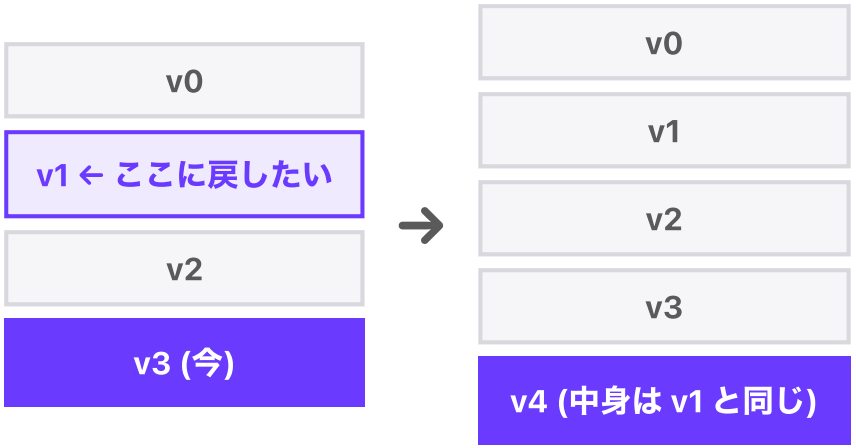
ai-desk: **かたまり (Block) = 過去の全バージョンの並び。**



「今のコード」は、**歴史の最後のページ**を読んでいるだけ。本体はページ全体。

5 歴史だから ― 戻すのも「消す」じゃない

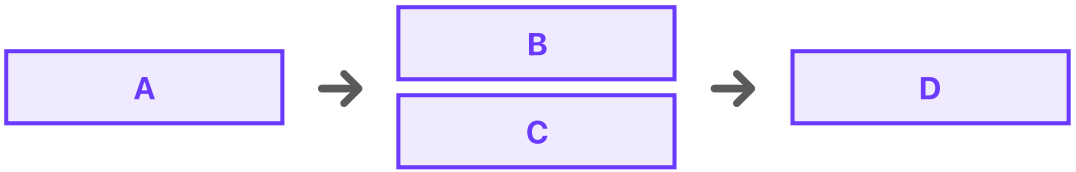
過去に戻りたい時、**過去を消す**のではなく **「あの時のページ」を新しく書き写す。**



失敗を恐れずに何度でも往復できる。過去が残ってるから、
「やり直し」が **新しい一歩** になる(削除でなく追加)。

6 発想の転換 ② — 関係も「明示」する

どの Block が、どの Block を呼んでいるか。
コードを読まなくても辿れる地図を持つ。



- **「A を変えたら何が壊れる?」** → 地図を辿れば一発で出る
- **「このバグの原因はどこ?」** → 逆方向に辿る
- **「過去のある時点の地図は?」** → 履歴と組合せで再現

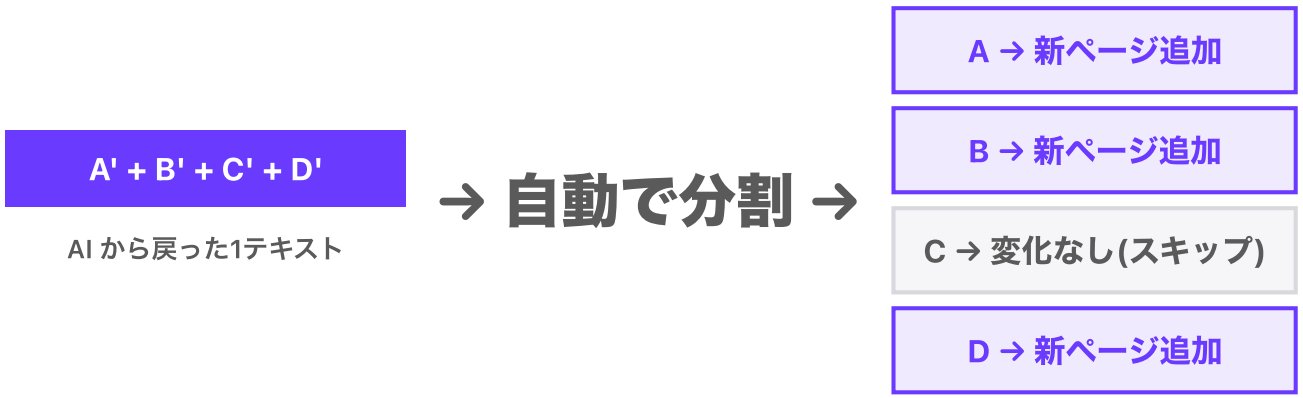
7 キラー機能 — 関係する全部をまとめて AI に渡す

「A を直して」と頼む時に、A + 呼ぶ先の B, C, D も一緒に同じプロンプトに展開する。

AI は **呼び出し元と呼ばれる側を同時に**見られる → 整合がとれた状態で直せる。

8 そして自動で配り直す — ここが本丸

AI から戻ってきた1テキストを、**機械的に元の各 Block に振り分けて**履歴に追加する。



1 回の AI 編集が、関連する Block 全てに同時に反映される。
「A 直したら B が古いまま壊れた」が **構造的に起きない**。

9 ３つの仕掛けが噛み合っている

1

かたまり (Block)

関数も、クラスも、ルールも、すべて同じ単位で扱う。AI は**1 種類のものとして読める**。

2

歴史 (Versions)

各 Block が自分の履歴を持つ。**失敗しても戻れる**から、AI に大胆に編集させられる。

3

地図 (Refs)

誰が誰を呼んでいるか明示。**「関係する全部」をまとめて**AI に渡せる。

この３つが組み合わさると

「AI に大きな修正を任せても崩壊しない」という挙動になる。

10 タグと優先順位 — AI が「読むべきところ」を絞れる

各 Block に **タグ**(例: `#high` `#logic` `#io`)と **優先順位**がつく。
AI は「**その中から検索**して必要な分だけ読む」。

巨大なコードベースでも、AI の視野は **絞れる**。
「全部読んで把握して」の幻覚モードを **呼び出さなくて済む**。

11 SPEC タグ — 仕様書が Block の中に住む

特殊タグ `#SPEC` がついた version は、**仕様だけ**を書いた一枚のページ。
初期仕様も、変更履歴も、Block の歴史にそのまま残る。

> 「この関数は**何をするはずだった**のか?」 → Block の歴史を見ればわかる。
> **仕様 と 実装 と 変更経緯**が同じ場所にある。外部ドキュメントが陳腐化する問題が消える。

12 git の代わりじゃない。git の上に乗る

GIT の役割

- ファイル全体の履歴
- 人がチームで開発するため
- ブランチ・マージ・PR

AI-DESK の役割

- **関数単位**の履歴と地図
- AI が編集するため
- 「関係する全部」をまとめる仕組み

両方使える。git は 保管庫、ai-desk は AI が触る作業台。

13 動いている — これは思想じゃなく実装

105
テスト全パス

4
単一HTMLで動くデモ
(ゲーム/表計算/ノードグラフ/カンバン)

0
外部ライブラリ
(Node 標準だけで動く)

どれも 1 セッションで AI が書いた。
これ自体が「崩壊しない」ことの証拠になっている。

歴史 + 地図 + まとめて渡す

この 3 つで、AI コーディングは
「関数だけ直して周りが壊れる」から抜け出せる。

新しい言語も、ビルドも、依存もいらない。
渡し方を変えるだけ。

github.com/AoyamaRito/ai-desk

沖井広行 · 蒼山りと