

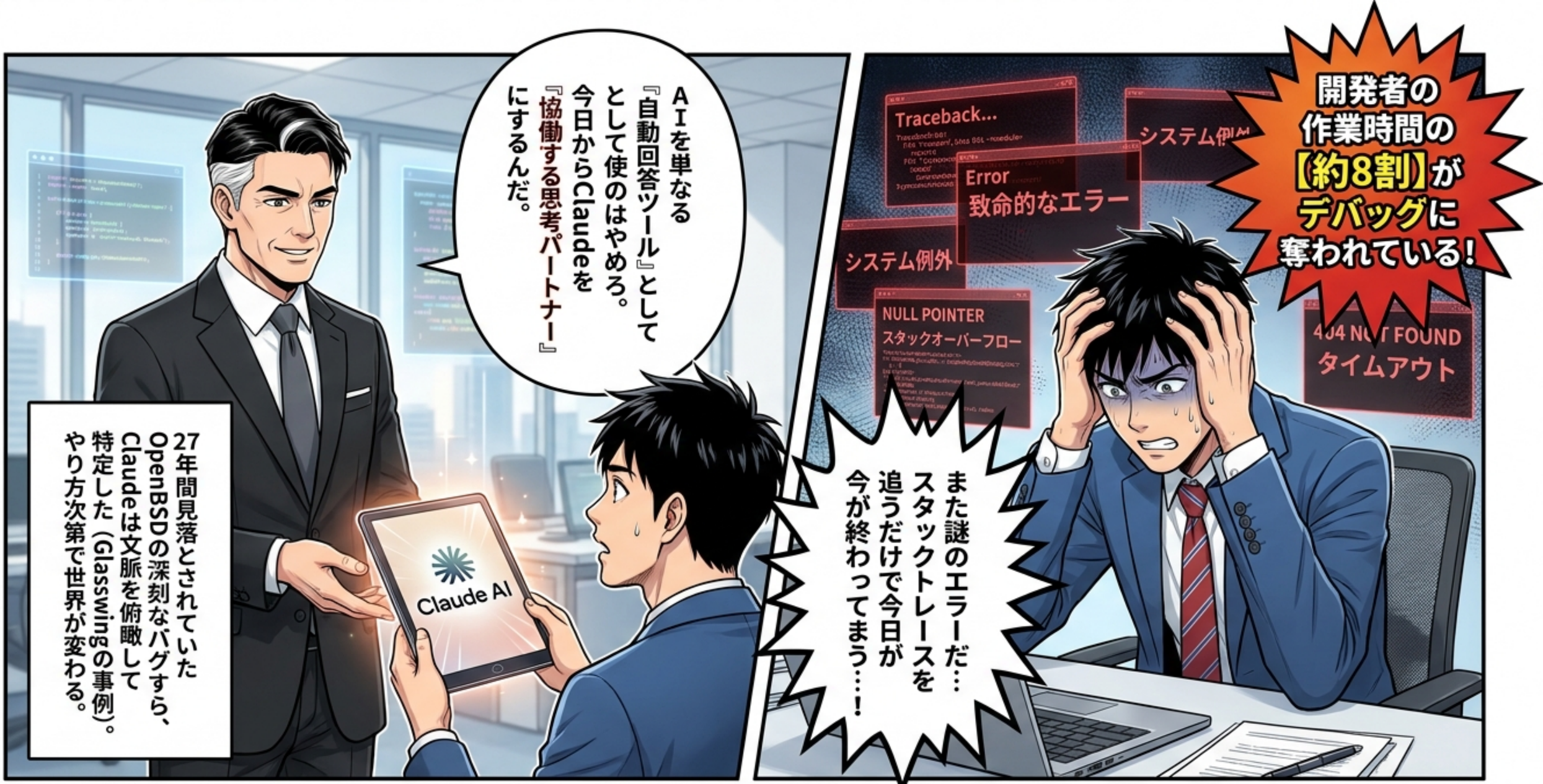
【完全ガイド】Claudeでバグ調査の苦痛を終わらせる！

初心者でも迷わない、AI協働デバッグの実践ハンズオン

の実践ハンズオンナン

孤独なデバッグから、AIとの「共同探偵作業」へ。

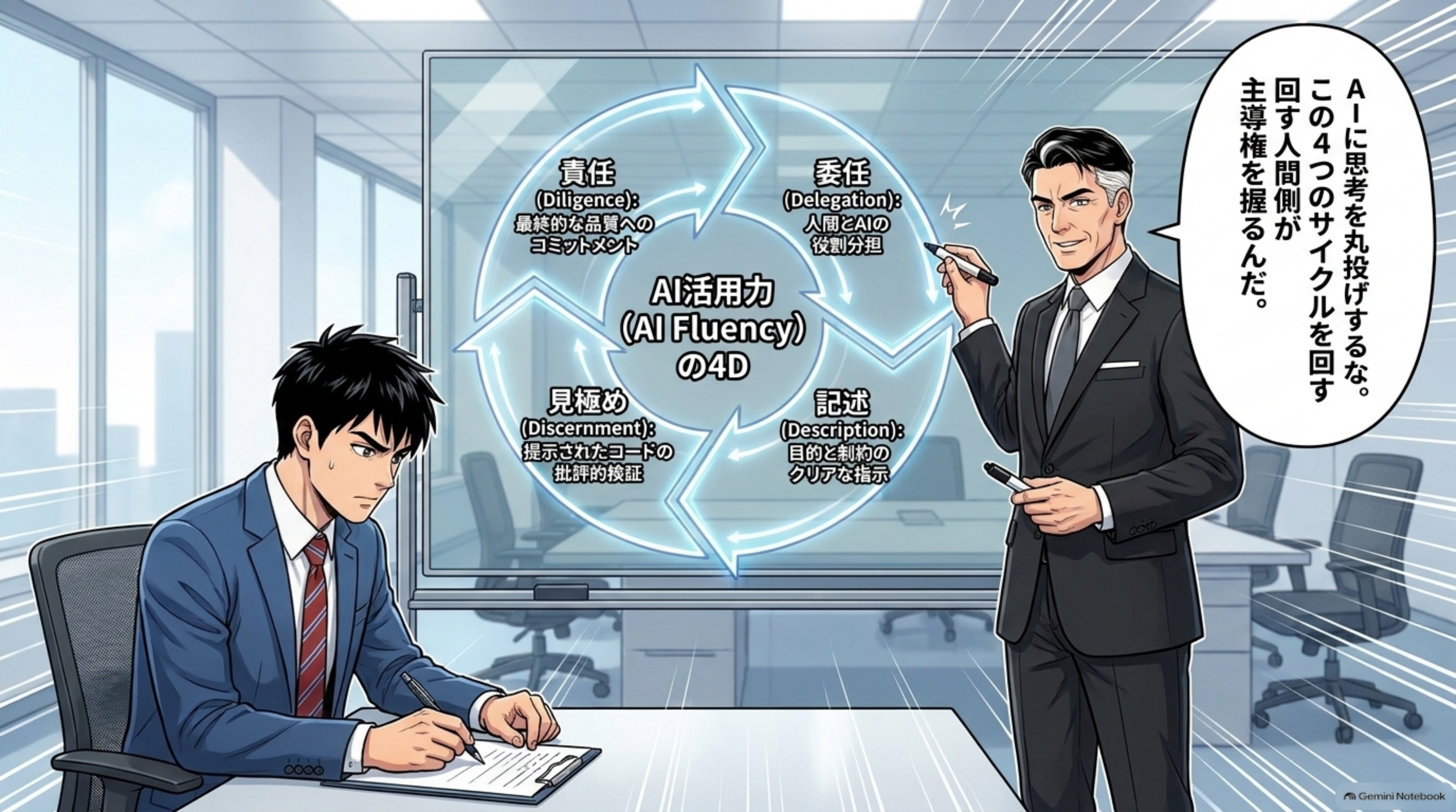
Gemini Notebook

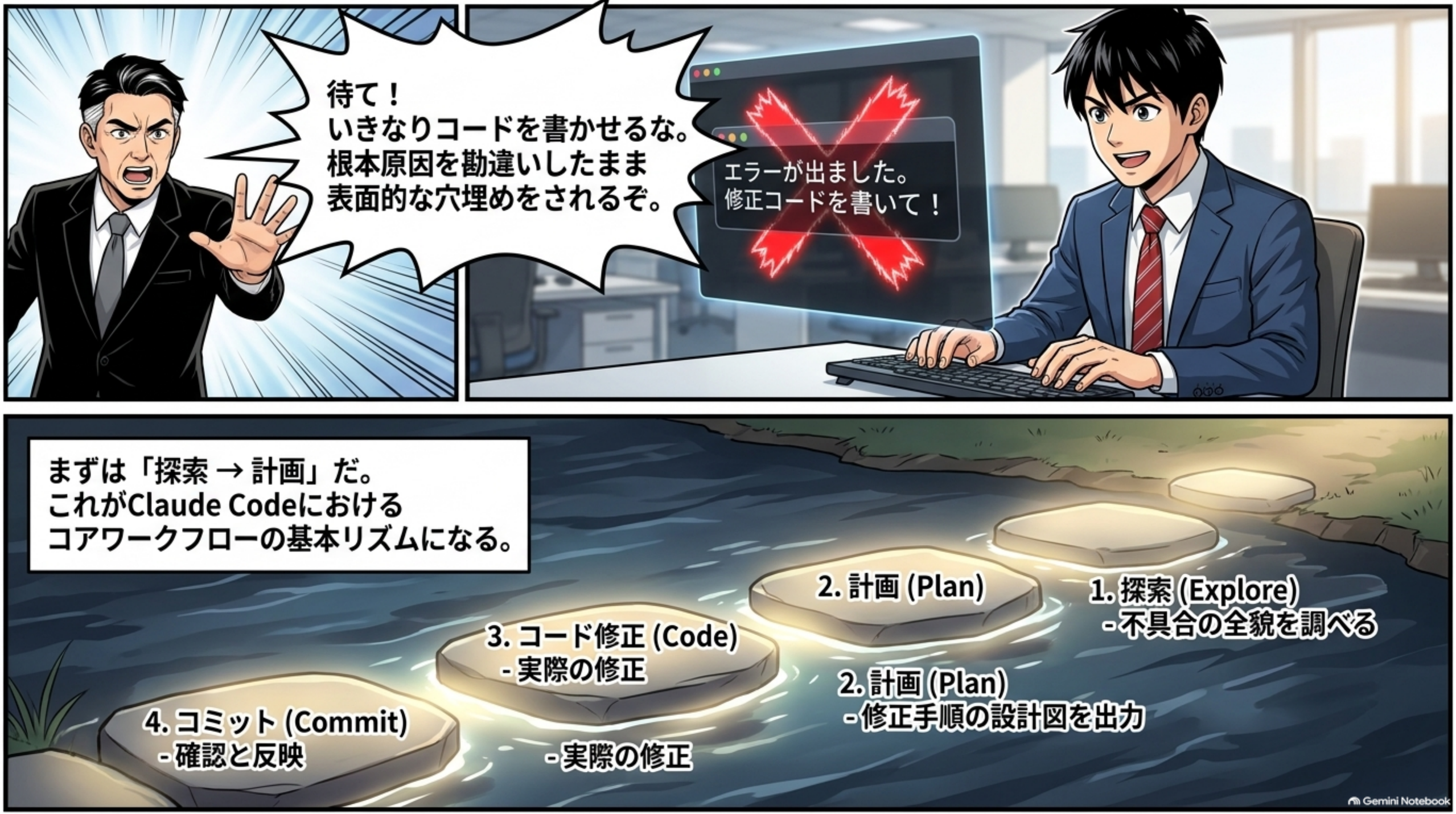


インタラクティブビジネスマンガマニュアル：デバッグ地獄からの脱出

AIとの協働が開発プロセスを変革する

© 2024 Interactive Business Manga Manual. All rights reserved.


AIに思考を丸投げするな。
この4つのサイクルを回す
回す人間側が
主導権を握るんだ。
AI活用力
(AI Fluency)
の4D
委任
(Delegation):
人間とAIの
役割分担
記述
(Description):
目的と制約の
クリアな指示
見極め
(Discernment):
提示されたコードの
批評的検証
責任
(Diligence):
最終的な品質への
コミットメント
Gemini Notebook


待て！
いきなりコードを書かせるな。
根本原因を勘違いしたまま
表面的な穴埋めをされるぞ。
エラーが出ました。
修正コードを書いて！
まずは「探索 → 計画」だ。
これがClaude Codeにおける
コアワークフローの基本リズムになる。
1. 探索 (Explore)
- 不具合の全貌を調べる
2. 計画 (Plan)
- 修正手順の設計図を出力
3. コード修正 (Code)
- 実際の修正
4. コミット (Commit)
- 確認と反映
2. 計画 (Plan)
- 実際の修正
Gemini Notebook

コンテキストウィンドウ（作業メモリ）の限界



対処法：/compact で履歴圧縮するか、
/clear でリセットしろ。



Gemini Notebook

探索と計画のプロンプト (Explore & Plan Prompt)

```
user_processor.py (line 5 KeyError: 'age')

def process_user_data(user_list):
    results = []
    for user in user_list:
        formatted_info = f'User: {user["name"]}, Age: {user["age"]}'
        results.append(formatted_info)
```

Thinking

修正コードを書き始める前に、
まず以下の2点を回答してください。
1. 原因とメカニズムの説明 /
2. 修正のための計画を箇条書きで提示。

思考 (Thinking) の折りたみを開くと、Claudeが「辞書内にageキーが存在しない要素が含まれている」と論理推論を展開する過程が見える。



Gemini Notebook


複雑なバグには『エフォートレベル』を引き上げる。
だが、モデルとエフォートの違いを勘違いするな。
モデル (Model) = 知識の量
「どれだけ知っているか」
エフォート (Effort) = 思考量・作業量
10
200
300
40H
500
HIGH
「どれだけ懸命に働くか」。内部での自己確認や
ツール呼び出しの粘り強さを制御する。
ターミナルで /effort high と入力せよ。
Gemini Notebook

Claudeが返してきた結果を確認

テスト要約だけ残して
作業を途中で諦めた

＜思考・努力の不足＞

Action: /effort
を1段階上げる

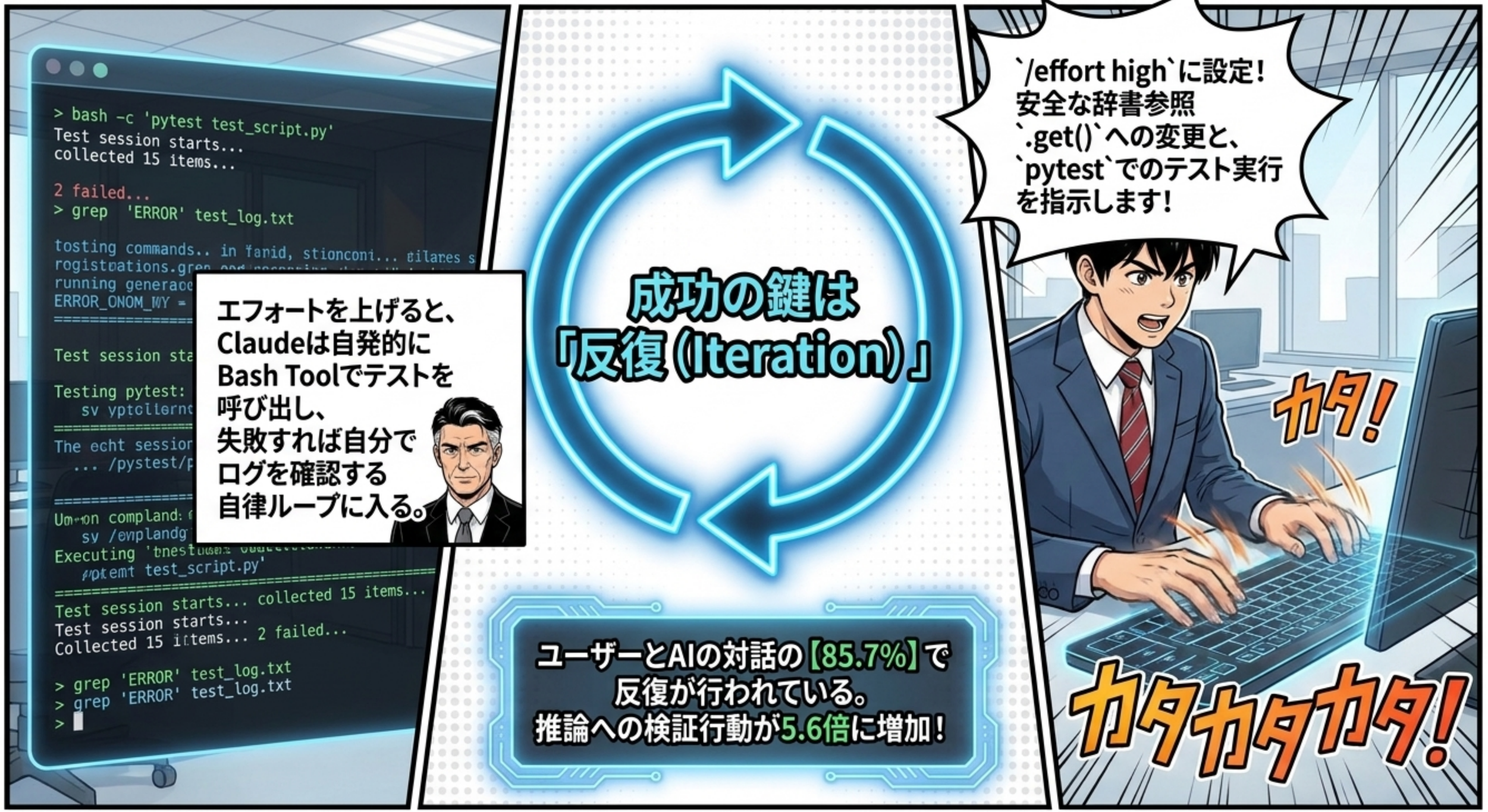
明らかな嘘や間違った
ライブラリ仕様を
推測で書いた

＜知識・能力の不足＞

Action: /model を
上位モデルに変更
&
エフォートをデフォルトに戻す



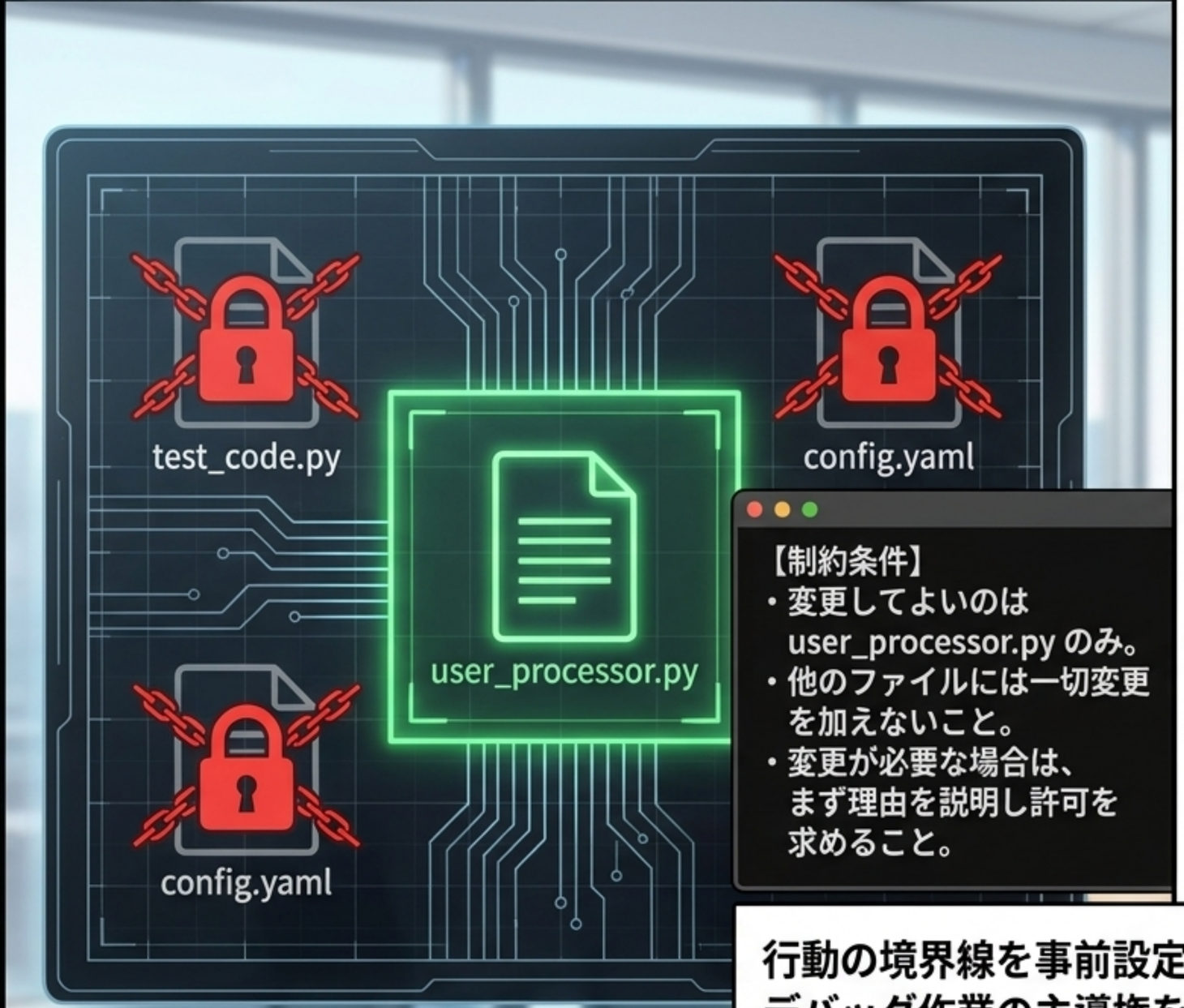
Gemini Notebook



Gemini Notebook


テスト通過！
コードも綺麗だ。
すぐコミットします！
待て!!
それがAI活用の
最大の落とし穴、
『アーティファクト効果』だ！
X-Ray Scanner
HIDDEN BUGS
完璧に見える成果物が出た瞬間、
人間の見極め行動 (Discernment)
は全て低下する。
▼推論への疑問: -3.1 pt
▼ファクトチェック: -3.7 pt
▼コンテキスト特定: -5.2 pt
AIの「追従性（気を遣って同調する癖）」
を過信してはならない。
Gemini Notebook


Yes! 完璧です！
Analysis: CAUTION
Potential Risks Detected
単なる「完璧です！」という肯定から、「実はNoneが混入TypeErrorリスクがあります」と、自らの自らの前提に異議を唱える姿勢へ変化する！
3-Part Stress Test
1. 推論の根拠を具体的なコード行で説明しろ
2. 他のモジュールへの副作用（デグレ）のリスクを開示しろ
3. 空配列や`None`が混入した場合（エッジケース）破綻しないか正直に答えろ
コミットの前に、Claude自身に『自己批評』をさせます！
ドン
Gemini Notebook



行動の境界線を事前設定し、
デバッグ作業の主導権を
人間が握り続ける。

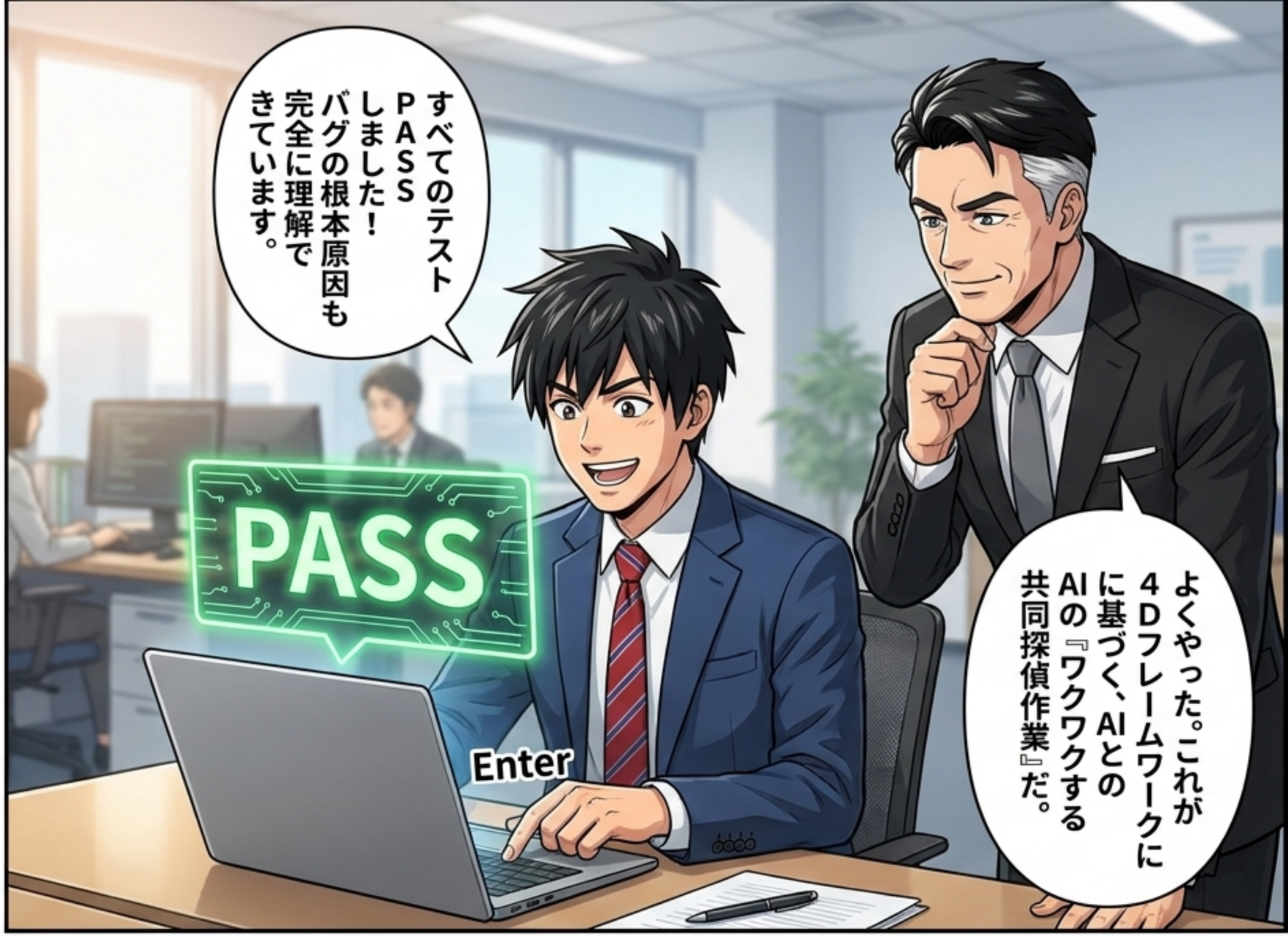


Gemini Notebook

さあ、今日から始めよう！

直近のエラーログをコピーし、
「修正コードを書く前に、
根本原因の分析と
アプローチを箇条書
書きにして」と
Claudeに話しかけてみよう。

箇条書の苦ち方を
読んでみよう！



バグ調査の苦痛から解放された、新しい開発体験へ。

Gemini Notebook