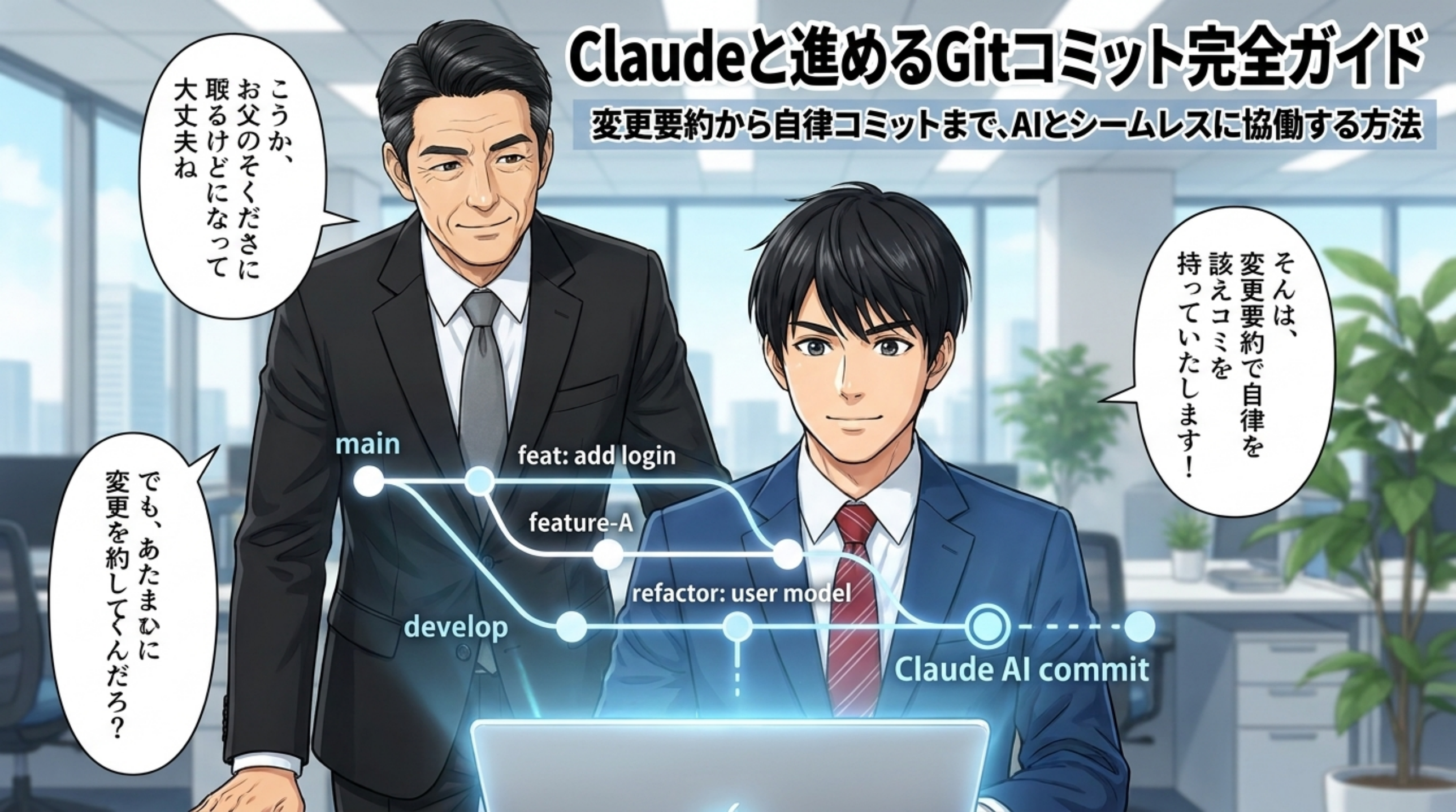
Claudeと進めるGitコミット完全ガイド

変更要約から自律コミットまで、AIとシームレスに協働する方法




せっかく集中していたのに、コミットメッセージの推敲で完全に手が止まってしまった…


その『コンテキストスイッチの摩擦』、強力なパートナーに委任（Delegation）して解消してみないか？


diff --git a/src/main.rs b/src/main.rs
--- a/src/main.rs
+++ b/src/main.rs
@@ -10,7 +10,7 @@
-    let config = load_config();
+   let config = match load_config() {
@@ -25,6 +25,9 @@
+   eprintln!("Error loading config: {}", e);
+   std::process::exit(1);
+   }
@@ -45,12 +48,15 @@
-    // Old logic for data processing
-    process_data(&data);
+   // New logic with enhanced error handling
+   if let Err(e) = process_data(&data) {
+   eprintln!("Processing failed: {}", e);
+   }
うわっ…複数のファイルを触りすぎて、どこをどう変えたか全体像が掴めない…

4D AI Fluency (AI活用力)



委任力 (Delegation)

・ルーチン的なコード差分の要約や文案作成を割り振る。



記述力 (Description)

・期待するコミット形式や文脈を正確に指示する。



評価力 (Discernment)

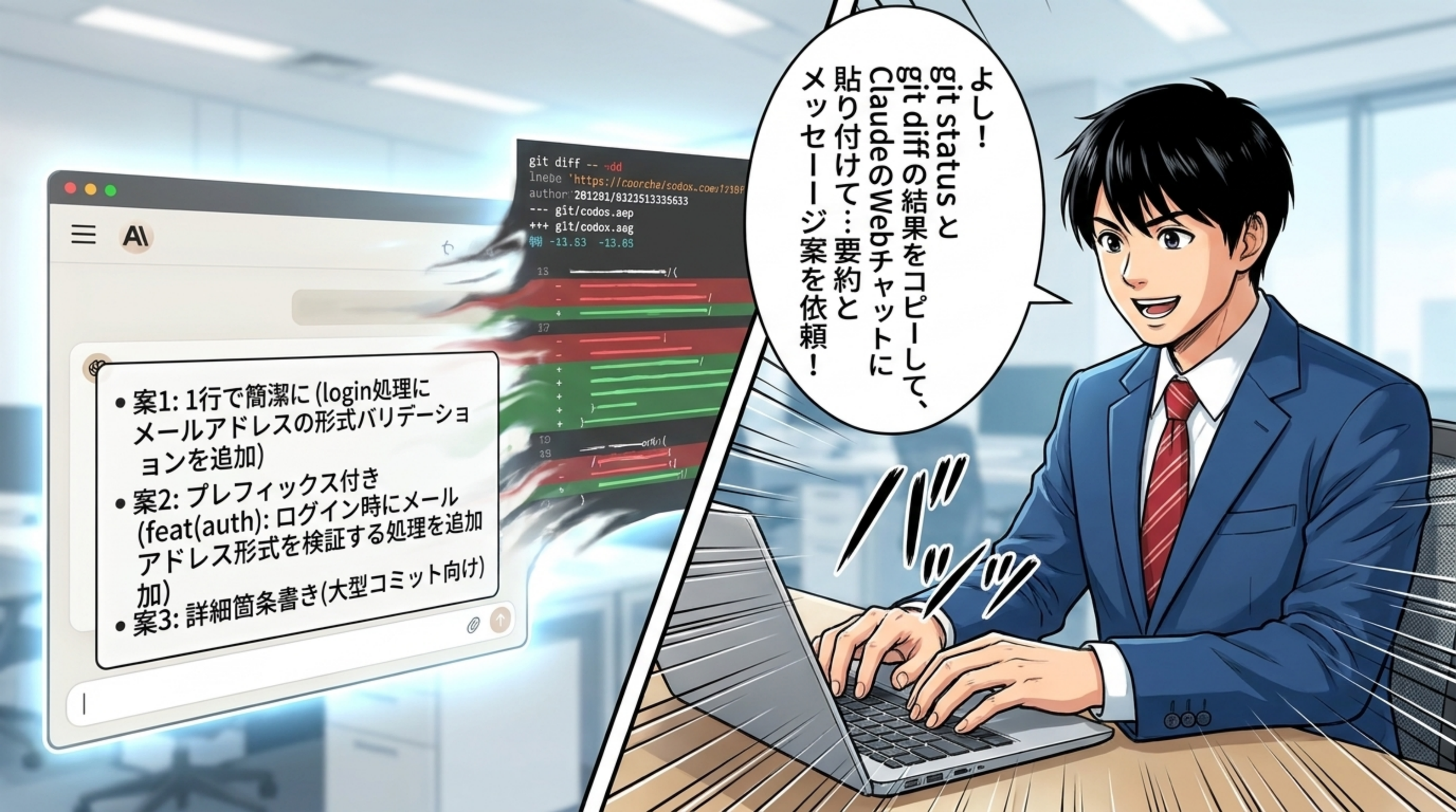
・ハルシネーション(作話)がないか人間が目視確認する。

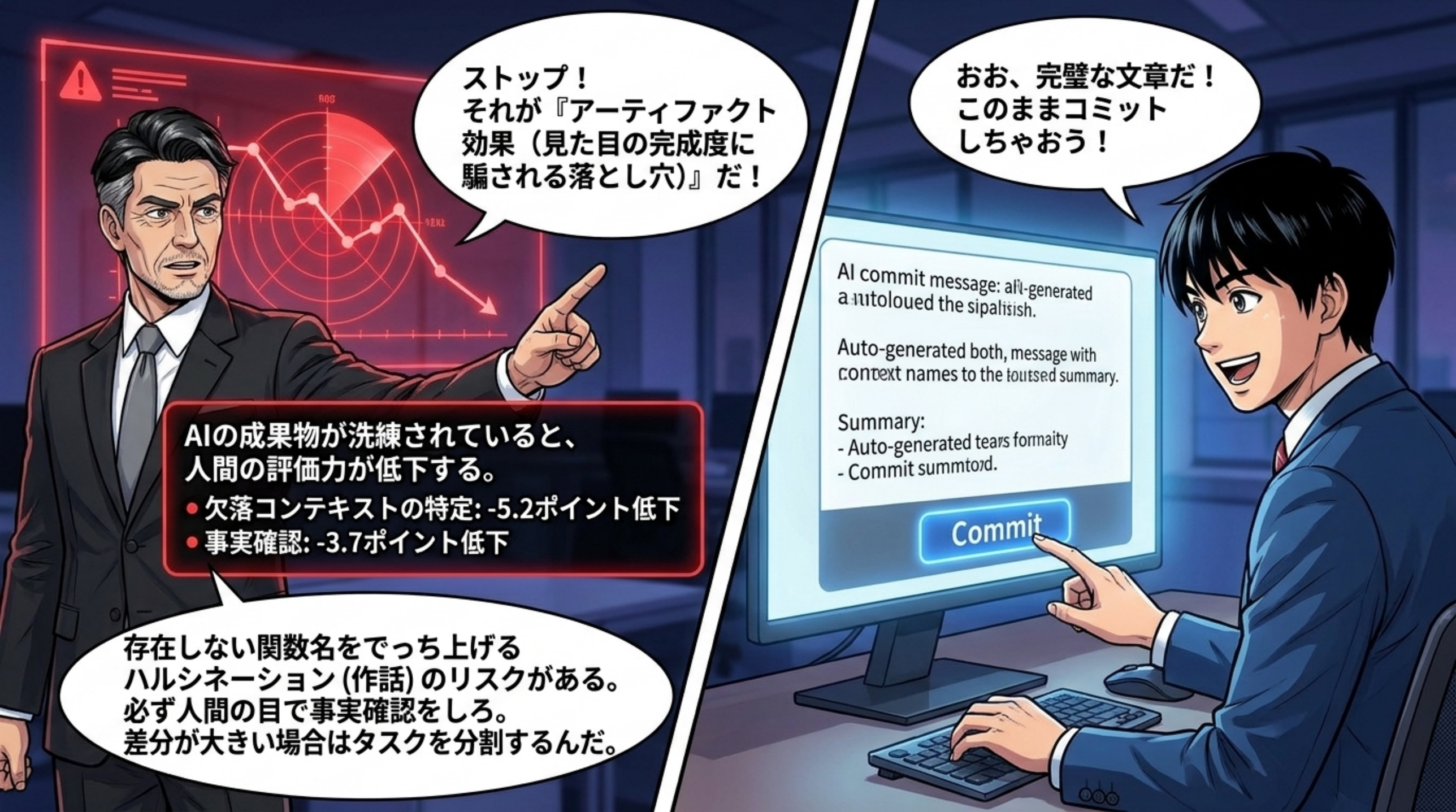


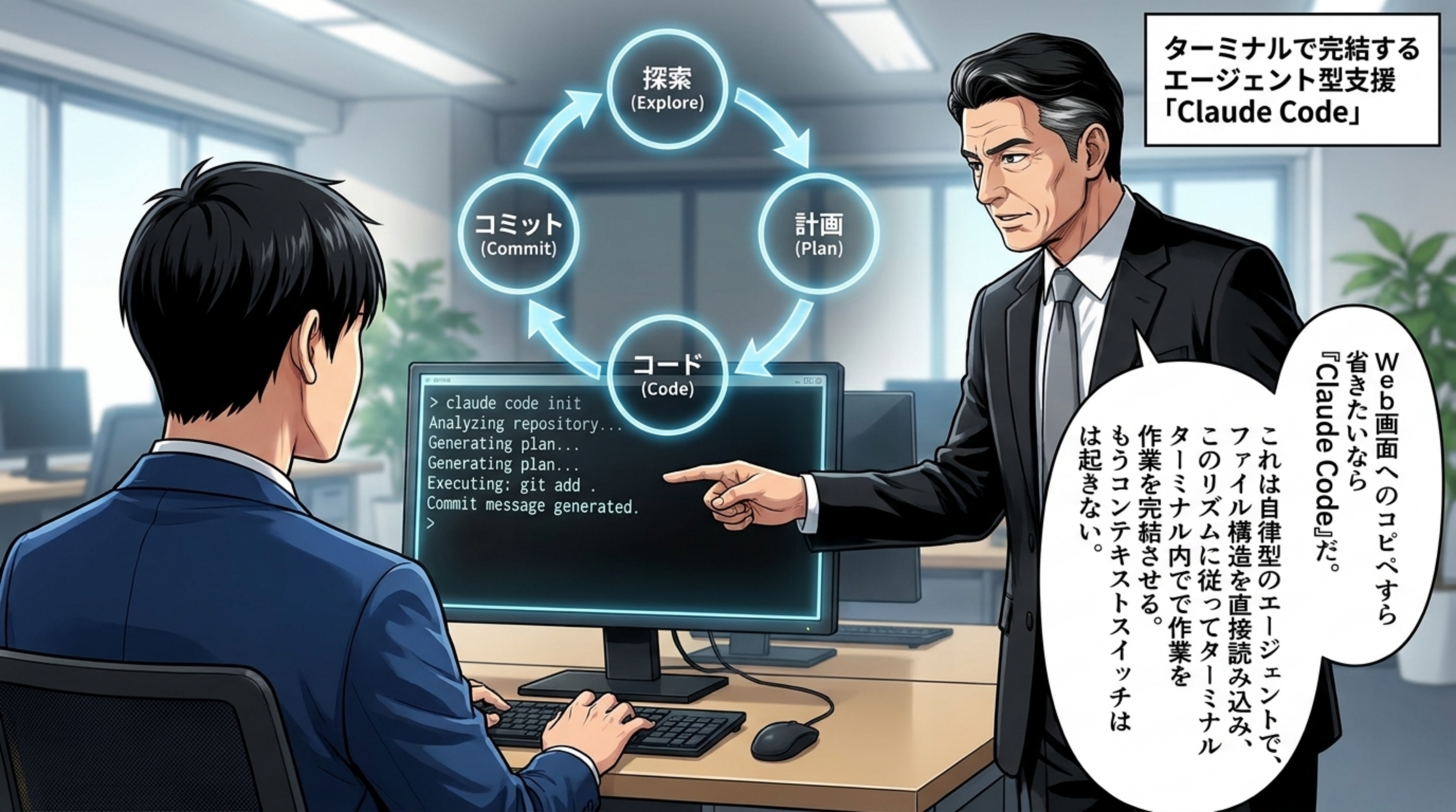
倫理的責任 (Diligence)

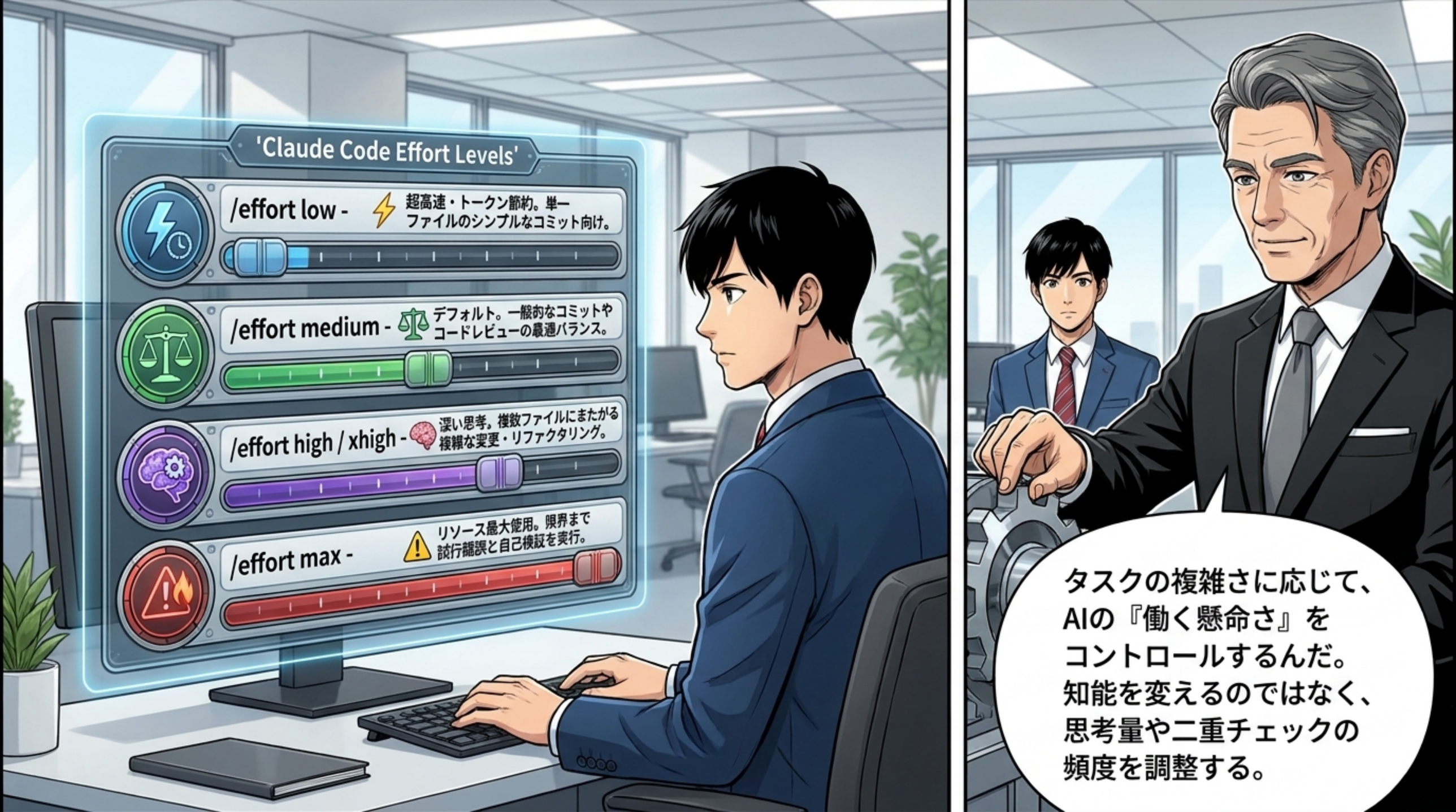
・履歴の正確性に最終的な責任を持つ。

AIと効果的に協働するための
世界標準フレームワーク、
『4D AI Fluency (AI活用力)』だ。
AIに任せることで、君は
『本質的な設計』に集中できるんだ。


よし！
git statusと
git diffの結果をコピーして、
Claudeのwebチャットに
貼り付けて…要約と
メッセージ案を依頼！
git diff -- add
--- git/codos.aep
+++ git/codox.aag
• 案1: 1行で簡潔に (login処理にメールアドレスの形式バリデーションを追加)
• 案2: プレフィックス付き (feat(auth): ログイン時にメールアドレス形式を検証する処理を追加)
• 案3: 詳細箇条書き(大型コミット向け)

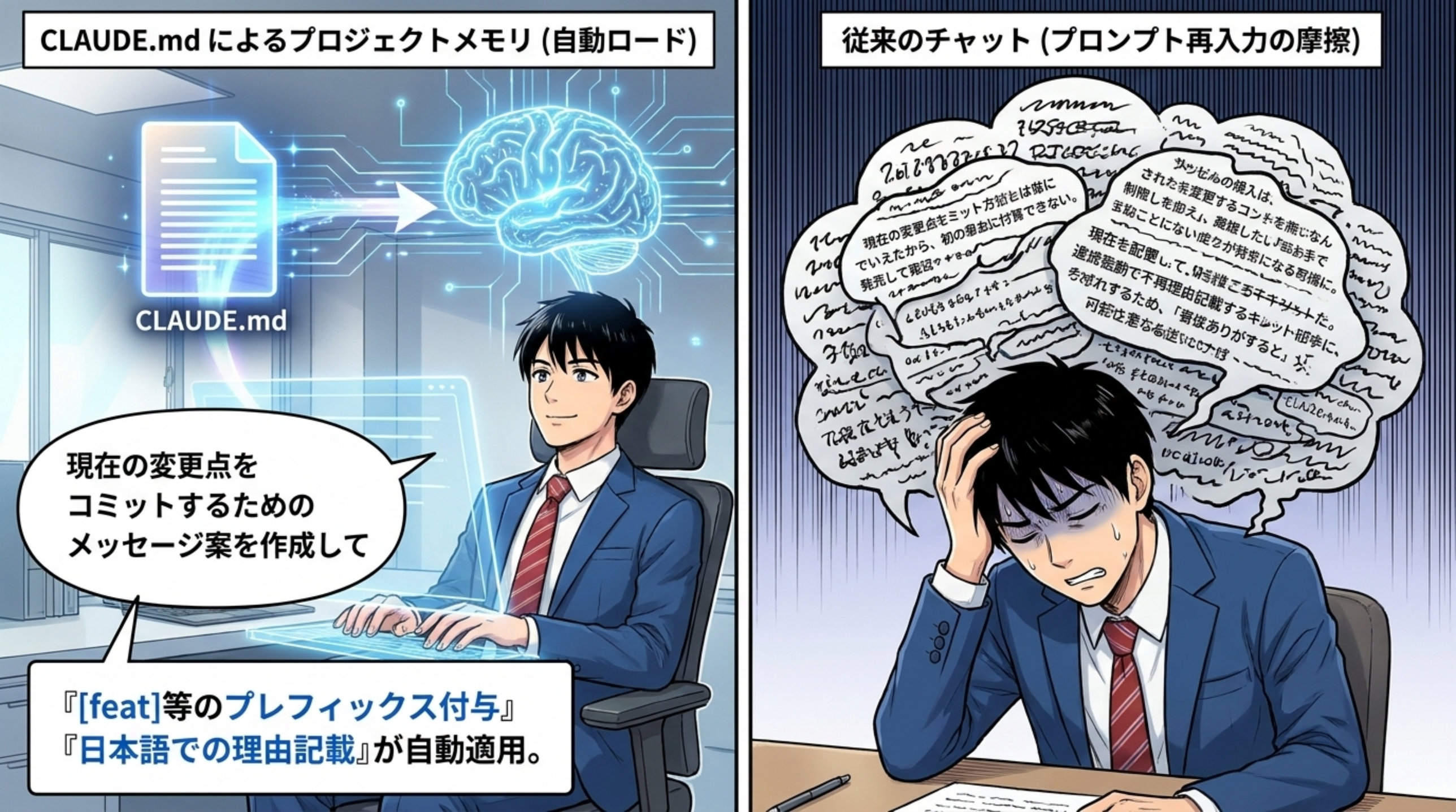

ストップ！
それが『アーティファクト効果（見た目の完成度に騙される落とし穴）』だ！
AIの成果物が洗練されていると、
人間の評価力が低下する。
• 欠落コンテキストの特定: -5.2ポイント低下
• 事実確認: -3.7ポイント低下
存在しない関数名をでっち上げる
ハルシネーション (作話) のリスクがある。
必ず人間の目で事実確認をしろ。
差分が大きい場合はタスクを分割するんだ。
おお、完璧な文章だ！
このままコミット
しちゃおう！
AI commit message: all-generated
autoloued the sipalisish.
Auto-generated both, message with
context names to the toutsed summary.
Summary:
- Auto-generated tears formaity
- Commit summtord.
Commit


ターミナルで完結する
エージェント型支援
「Claude Code」
探索
(Explore)
計画
(Plan)
コード
(Code)
コミット
(Commit)
> claude code init
Analyzing repository...
Generating plan...
Generating plan...
Executing: git add .
Commit message generated.
>
Web画面へのコピペすら
省きたいなら
『Claude Code』だ。
これは自律型のエージェントで、
ファイル構造を直接読み込み、
このリズムに従ってターミナル
作業を完結させる。
もうコンテキストスイッチは
は起きない。

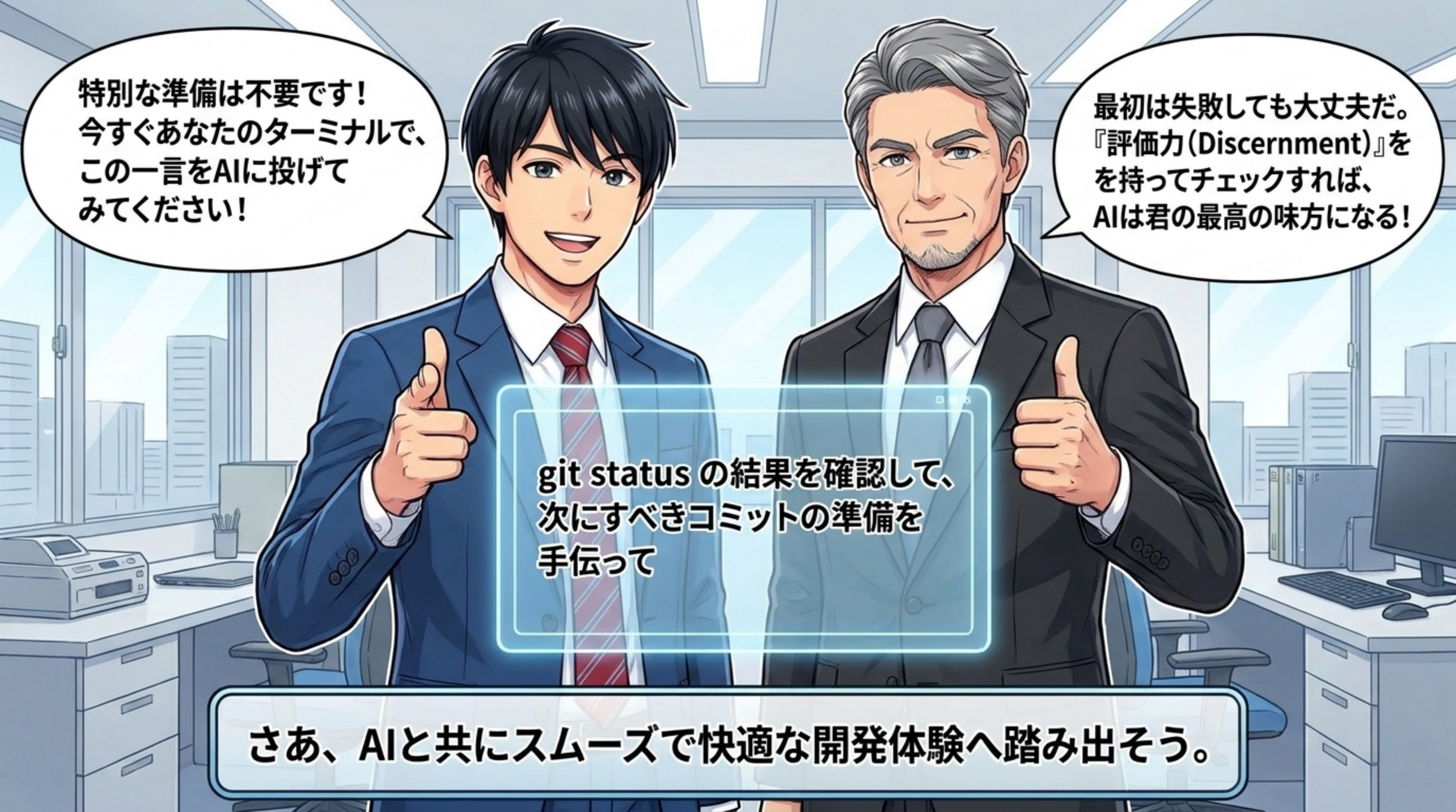

'Claude Code Effort Levels'
/effort low - 超高速・トークン節約。単一ファイルのシンプルなコミット向け。
/effort medium - デフォルト。一般的なコミットやコードレビューの最適バランス。
/effort high / xhigh - 深い思考。複数ファイルにまたがる複雑な変更・リファクタリング。
/effort max - リソース最大使用。限界まで試行錯誤と自己検証を実行。
タスクの複雑さに応じて、
AIの『働く懸命さ』を
コントロールするんだ。
知能を変えるのではなく、
思考量や二重チェックの
頻度を調整する。


AIは勝手に実行したりしない。 意図しない未完成ファイルのコミットを防ぐため、人間が必ず確認・承認（Diligence）するガードレールが機能しているんだ。
Do you want to run:
git commit -m "feat(auth): add email validation logic and unit tests"? (y/n)


指示が長すぎてモデルがルールを見落とす『推論のドリフト (Steering Drift)』だな。ならば、プロジェクト固有のルールを『永続記憶』させよう。
MD
ええと…『メッセージは日本語で』『[feat]等のプレフィックスをつけて』『Issue番号を含めて』…毎回このルールを書くのは面倒くさい!!しかもたまにAIが指示を忘れる(推論のドリフト)し!


CLAUDE.mdによるプロジェクトメモリ (自動ロード)
CLAUDE.md
現在の変更点を
コミットするための
メッセージ案を作成して
『[feat]等のプレフィックス付与』
『日本語での理由記載』が自動適用。
従来のチャット (プロンプト再入力の摩擦)


3 step 合成合法
1. Web UI ➡ 差分要約の委任
2. Claude Code ➡ ターミナル完結の自律動作
3. CLAUDE.md ➡ ルールの自動適用
コミット作成の摩擦から解放され、
開発者は最も価値ある『本質的なロジック構築』に100%集中できる。


特別な準備は不要です！
今すぐあなたのターミナルで、
この一言をAIに投げて
みてください！
最初は失敗しても大丈夫だ。
『評価力（Discernment）』を
を持ってチェックすれば、
AIは君の最高の味方になる！
git status の結果を確認して、
次にすべきコミットの準備を
手伝って
さあ、AIと共にスムーズで快適な開発体験へ踏み出そう。