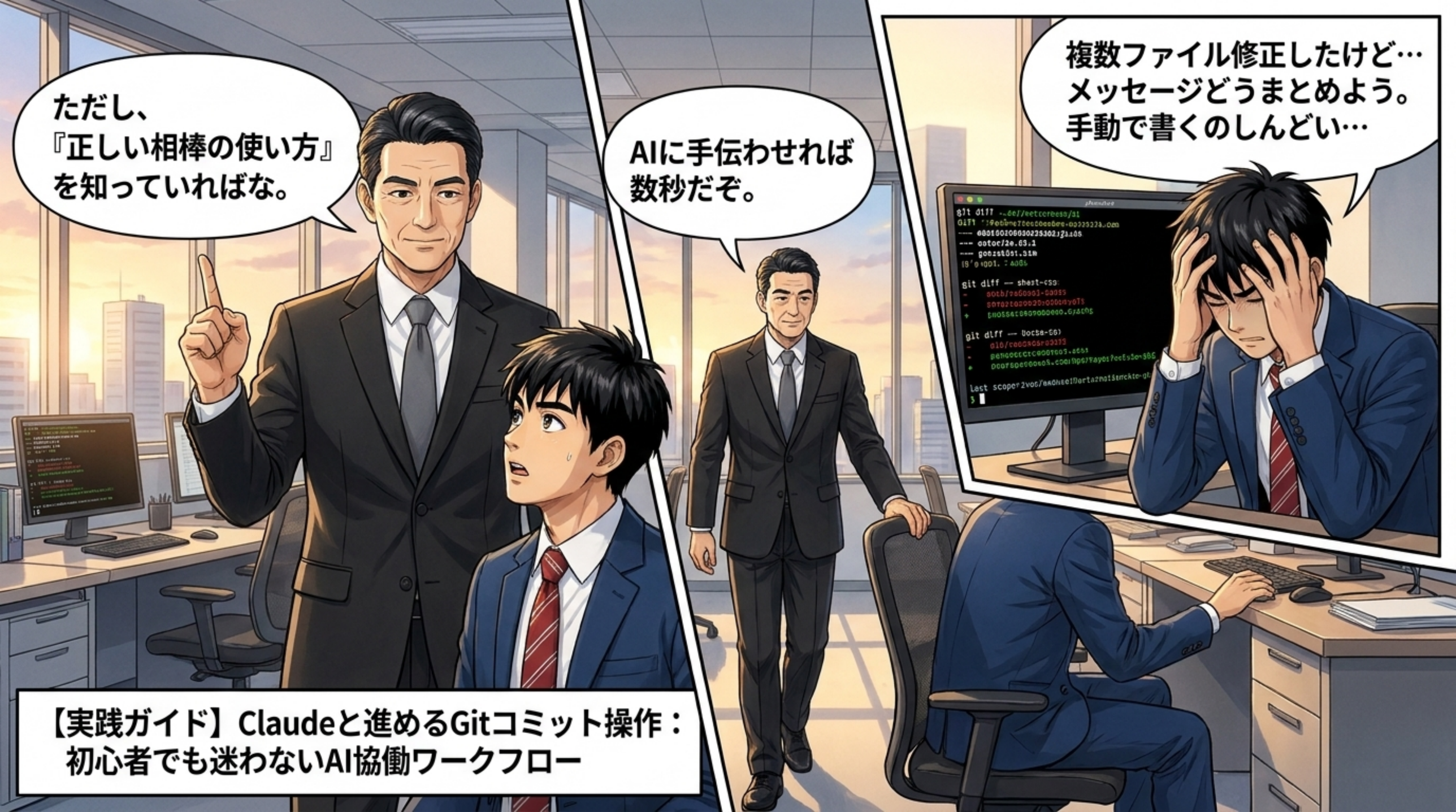
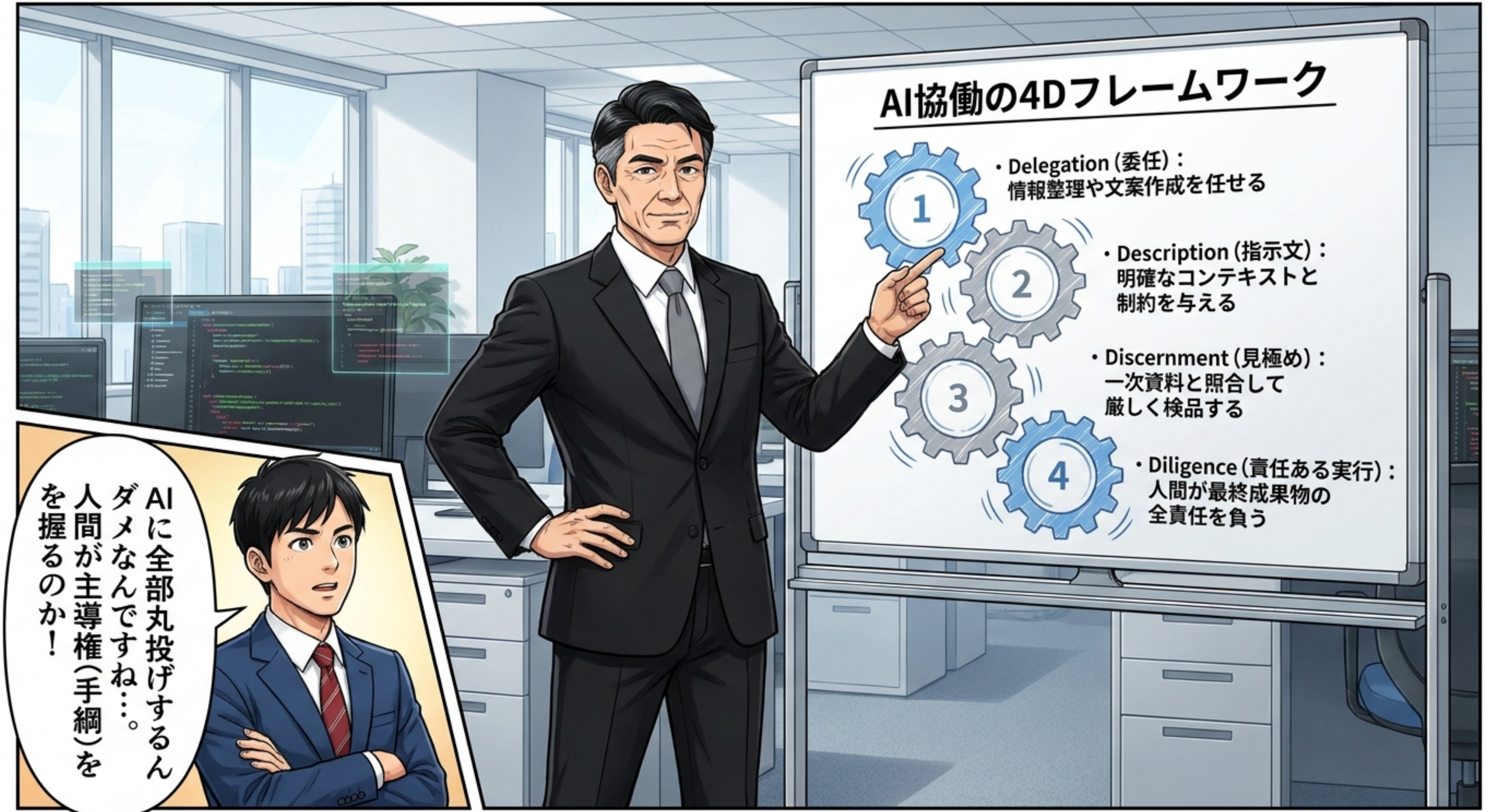
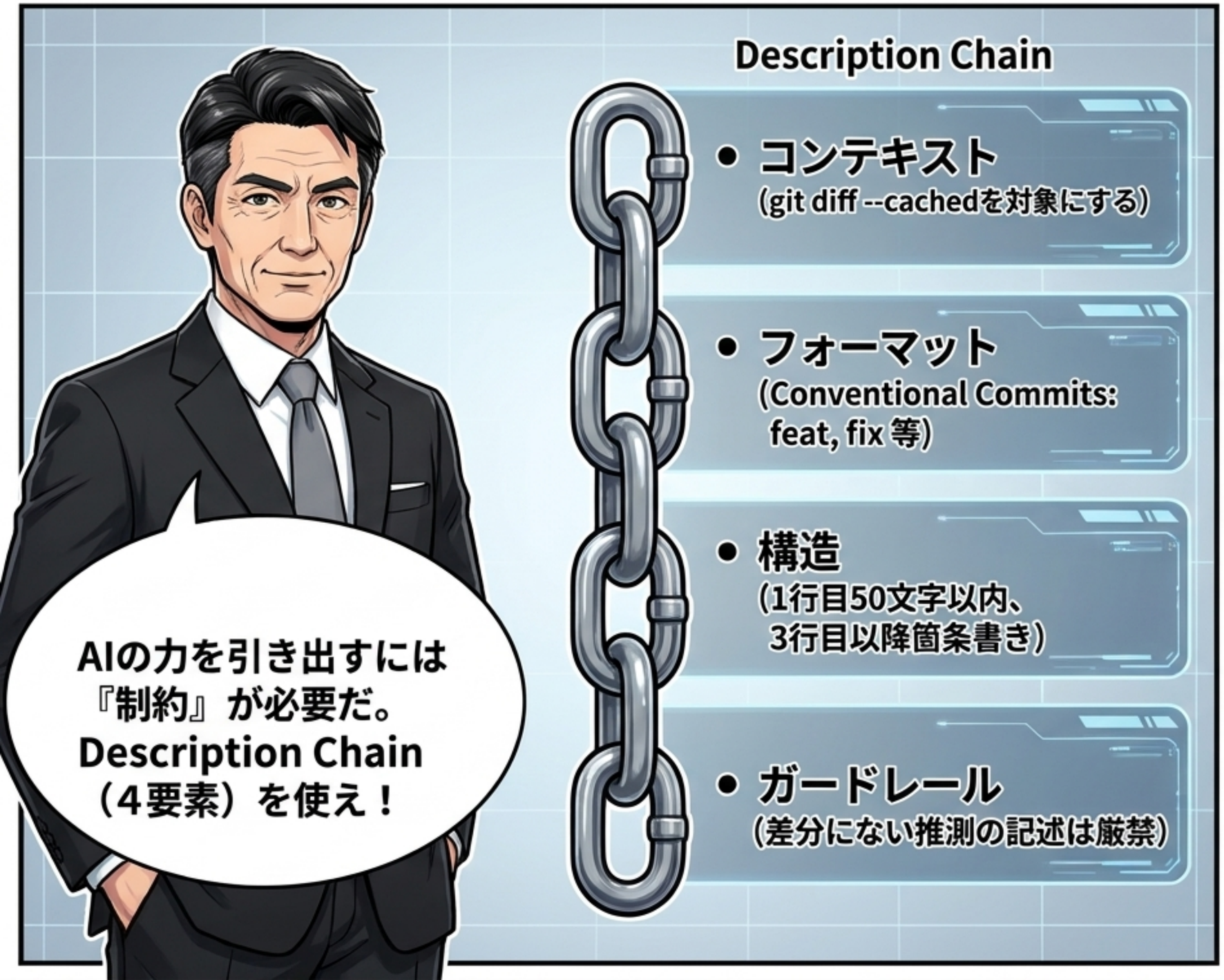

ただし、
『正しい相棒の使い方』
を知っていればな。
AIに手伝わせれば
数秒だぞ。
複数ファイル修正したけど…
メッセージどうまとめよう。
手動で書くのしんどい…
【実践ガイド】Claudeと進めるGitコミット操作：
初心者でも迷わないAI協働ワークフロー


AI協働の4Dフレームワーク
1
・Delegation（委任）：
情報整理や文案作成を任せる
2
・Description（指示文）：
明確なコンテキストと
制約を与える
3
・Discernment（見極め）：
一次資料と照合して
厳しく検品する
4
・Diligence（責任ある実行）：
人間が最終成果物の
全責任を負う
AIに全部丸投げするんダメなんですね…。人間が主導権（手綱）を握るのか！

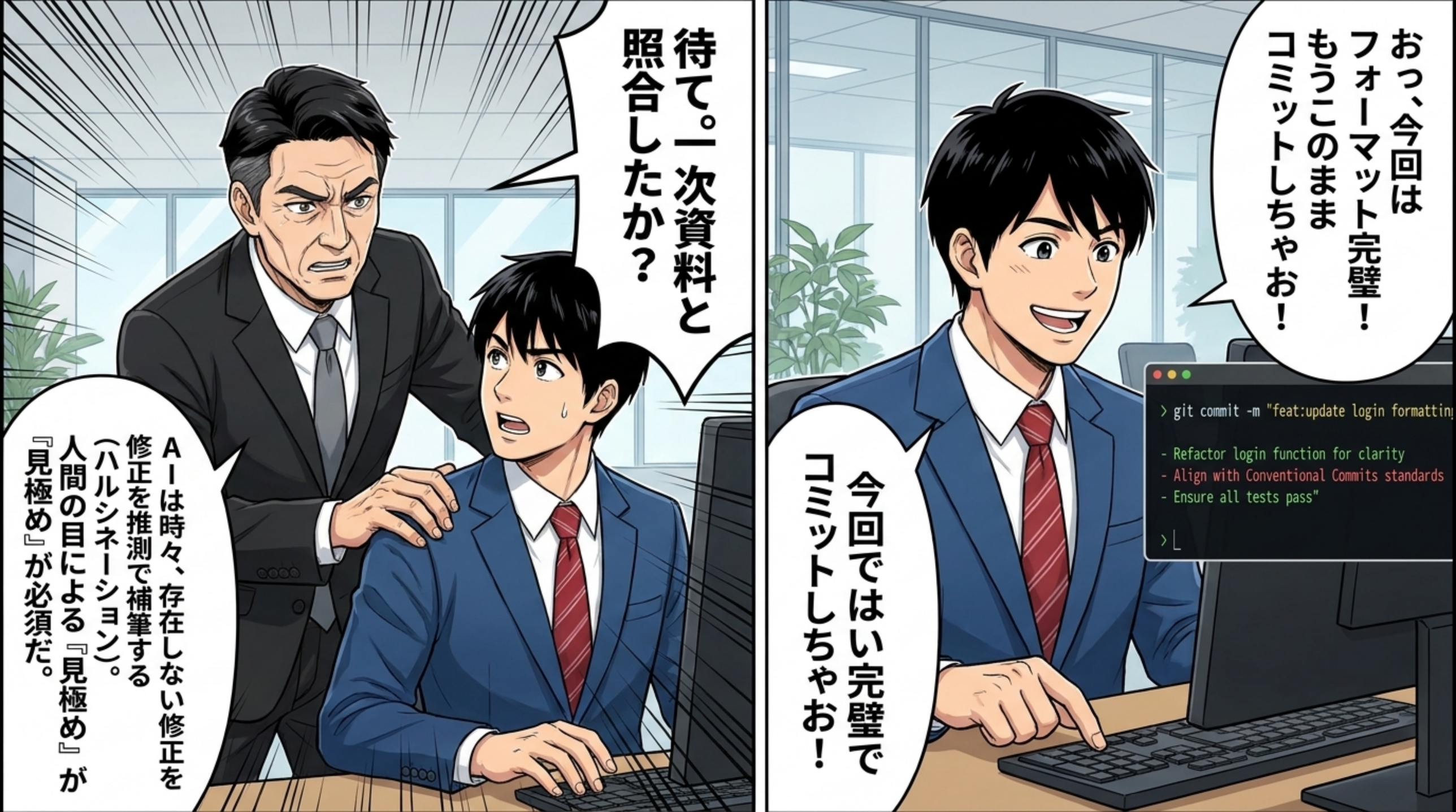

よし、1行のタイポ修正だけど、
完璧を期して
最大限頑張ってもらおう！
エフォートMAXだ！
タイポ修正
ゴタッ
えっ!? たった1行の修正
修正なのに全然わらいし、
謎のツールを
呼び出しまくってる!?
Thinking...
Thinking...
Thinking...
Thinking...
待て。エフォートは
『頭の良さ』じゃなく
簡単な作業にリソースを
無駄遣いするな！
バッ

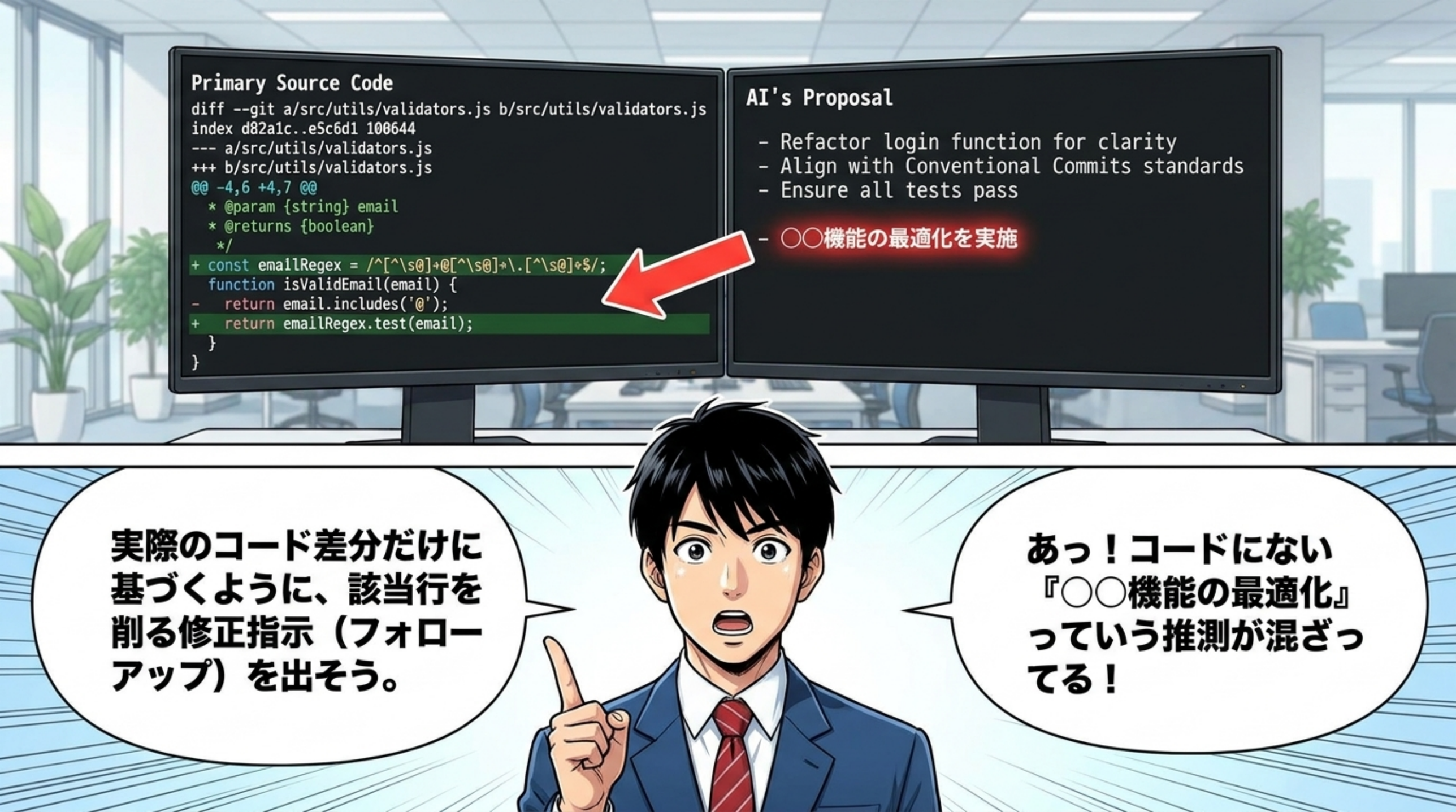

low（低コスト・超速）：
軽微な修正、タイポ補正。最小限の思考。
medium（デフォルト）：
通常の機能追加やバグ修正。
high / xhigh（高コスト・慎重）：
複数ファイルに跨るリファクタリング。
max（最大コスト）：
限界までの思考。※セッション限定。
LOW
MAX
作業の複雑さに合わせて、
AIの『働く努力量』のダイヤルを
ダイヤルを回すんだ。
常にデフォルトから始めろ。
> claude --effort low
軽微な作業だから、
まずはlowで
起動してみよう！

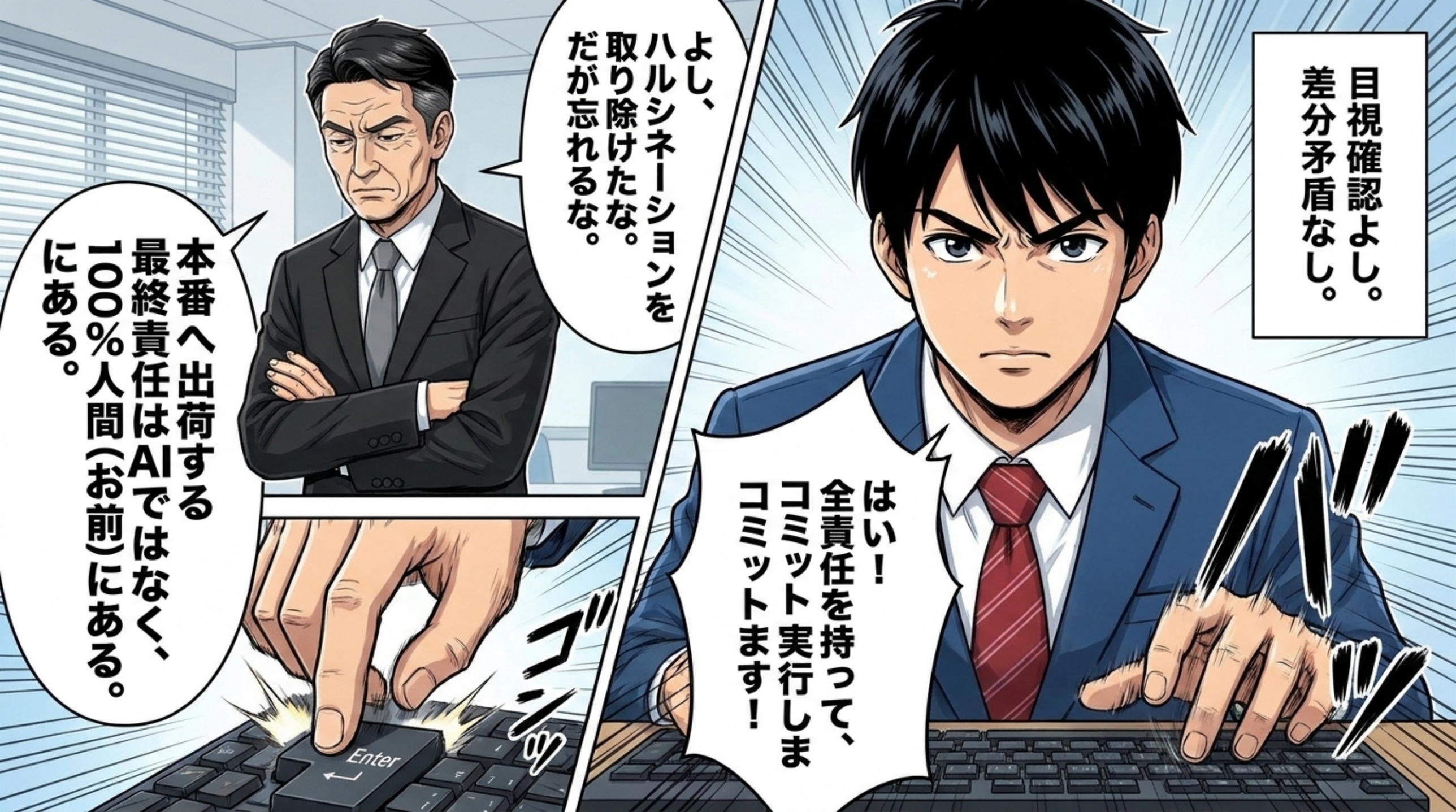

Description Chain
• コンテキスト
(git diff --cachedを対象にする)
• フォーマット
(Conventional Commits:
feat, fix 等)
• 構造
(1行目50文字以内、
3行目以降箇条書き)
• ガードレール
(差分にない推測の記述は厳禁)
AIの力を引き出すには
『制約』が必要だ。
Description Chain
（4要素）を使え！

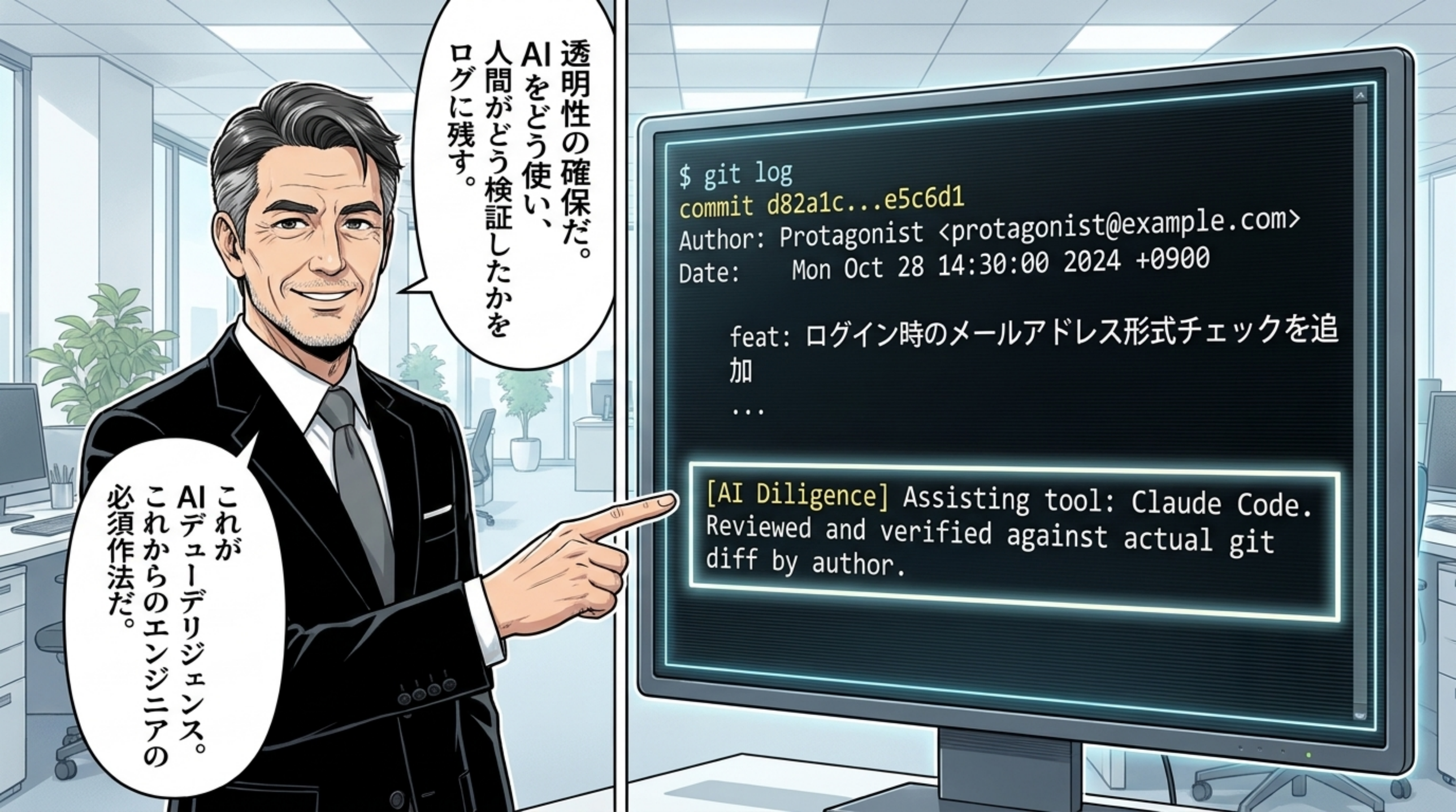

いい感じに
コミットメッセージ
書いて！


ログイン機能が美しく
生まれ変わりました✨
完璧なる検証の旅へ…
ゴダッ


おっ、今回はフォーマット完璧！もうこのままコミットしちゃお！
> git commit -m "feat:update login formattin
- Refactor login function for clarity
- Align with Conventional Commits standards
- Ensure all tests pass"
今回ではい完璧でコミットしちゃお！
待て。一次資料と照合したか？
AIは時々、存在しない修正を修正を推測で補筆する（ハルシネーション）。人間の目による『見極め』が『見極め』が必須だ。


Primary Source Code
diff --git a/src/utils/validators.js b/src/utils/validators.js
index d82a1c..e5c6d1 100644
--- a/src/utils/validators.js
+++ b/src/utils/validators.js
@@ -4,6 +4,7 @@
* @param {string} email
* @returns {boolean}
*/
+ const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
function isValidEmail(email) {
- return email.includes('@');
+ return emailRegex.test(email);
}
AI's Proposal
- Refactor login function for clarity
- Align with Conventional Commits standards
- Ensure all tests pass
- ○○機能の最適化を実施
実際のコード差分だけに基づくように、該当行を削る修正指示（フォローアップ）を出そう。
あっ！コードにない『○○機能の最適化』っていう推測が混ざってる！


目視確認よし。
差分矛盾なし。
はい！
全責任を持って、
コミット 実行しま
コミットます！
よし、
ハルシネーションを
取り除けたな。
だが忘れるな。
本番へ出荷する
最終責任はAIではなく、
100%人間（お前）にある。
ゴッ
Enter


透明性の確保だ。
AIをどう使い、
人間がどう検証したかを
ログに残す。
これが
AIデューデリジェンス。
これからのエンジニアの
必須作法だ。
$ git log
commit d82a1c...e5c6d1
Author: Protagonist <protagonist@example.com>
Date: Mon Oct 28 14:30:00 2024 +0900
feat: ログイン時のメールアドレス形式チェックを追加
...
[AI Diligence] Assisting tool: Claude Code.
Reviewed and verified against actual git diff by author.


まずは手元の小さな修正で、
`claude --effort low`から
試してみろ。
AIは優秀な『相棒』になるぞ。
4Dフレームワークがあれば、
もうGitコミットで迷わない！
❯ claude --effort low
❯ このリポジトリの最新のGit変更点を確認し、要約してください