Lightning Talk

AI活用のコスパを最大化する方法

トークン制約時代の 依頼設計・CLI運用・共有資産化

対象: AI活用を始めたエンジニア / テックリード

2026.02.19 LT

猫とゲームを愛するエンジニア

最近X投稿さぼってる＼(^o^)／

IT関連の情報発信（とゲームや日常をポスト）
よかったらフォローしてください!!

X：@ochtum18
Github：ochtum
Qiita：ochtum
SpeakerDeck：ochtum

今日のゴール

「再試行を減らす設計」で成果・速度・コストを同時改善する

1. 依頼設計

目的・前提・制約・期待出力を設計し、最初の1回で精度を上げる。

2. CLI運用

CLI活用で高パフォーマンスの運用設計をする。

3. 共有資産化

テンプレ・チェックリスト・スキルを資産化し、個人最適をチーム最適へ。

なぜコスパ設計が重要か

AI活用のコスパ = 単価ではなく「完了までの総コスト」

曖昧な依頼

方向ズレ→追加指示→手戻り増大。結果としてトークン消費が増える。

構造化した依頼

1回目成功率が上がり、往復と認知負荷を同時に削減できる。

評価指標（まず3つ）

- 完了までのやり取り回数
- 1タスクあたりの所要時間
- レビュー差し戻し件数

総コストが下がれば、実質的なコスパは大きく向上する

1. 依頼設計

最初に避けるべき失敗パターン

要件が1文

「いい感じで直して」では優先順位が不明。AI出力がぶれやすい。

制約が未記載

禁止事項がないと後出し修正が連鎖する。

依頼が巨大

調査・設計・実装・検証を一括で投げると精度が落ちる。

“良い文章”よりも“良い構造化”

“良い文章”に時間をかけるより、“良い構造化”に時間をかけたほうがランニングコストが下がる＼(^o^)／

・セクション(#)を分ける
・実施フローを伝える
・ルール/詳細説明などのファイルを明確に分ける
・変数を使う(AIはプレースホルダと解釈する）
・スクリプトを実行する
・テンプレートを用意する
・あえて使うツールやスキルを指定する
・実行単位を分ける

伝わる依頼テンプレートの例（Markdown固定）

```
# 目的
- 何を達成したいか

# 前提
- 現在の状況
- 使用環境

# 制約
- 使ってよい技術 / 禁止事項
- 納期・優先順位

# 期待する出力
- 形式（Markdown / JSON / コード）
- 完了条件
```

自由記述より構造化。これだけで初回品質は安定する。

複雑依頼は4フェーズ分割

1回で完璧を狙わない。分割で精度と速度を上げる。

1. 調査

用語定義
前提確認

2. 設計

構成案
採用/非採用理由

3. 実装

本文/コード生成
形式を統一

4. レビュー

抜け漏れ
要件充足を検証

分割すると「どこで品質が落ちたか」を切り分けやすい。

2. CLI運用

GUIよりCLIを選ぶべき理由

AIペアプログラマーとして使わないならば単純にCLIが良い。

- ⚡ パフォーマンスがよい場合が多い
- 🎯 成果がよいことが多い
- 💰 消費トークン数が少ない場合がある
- 🙌 自分の作業の裏で動かせる（マルチタスク化）

※実務観測ベースの傾向。CLIは「邪魔しない導線」も利点。

3. 共有資産化

チームで効く共有資産設計

個人の成功パターンを、再利用可能な資産に変える。

入力資産

依頼テンプレ
レビュー観点
検証手順

運用資産

スクリプト(AIが実行)
カスタムエージェント
カスタムプロンプト
スキル

組織効果

品質の底上げ
オンボード短縮(早期戦力化)
属人化の解消

４．補足

セキュリティ前提の運用ルール

無料版を併用するなら・・・

必須ルール

- 個人情報・顧客情報・機密情報は入力しない
- 必要時はマスキングしてから投入する
- ログ保存範囲と公開範囲を事前確認する
- 社内ガイドラインに沿って手順を明文化する

よくある詰まり

・ルールが多すぎて使われない

・更新されず古くなる

・担当者依存でブラックボックス化

対策: 最小セットで開始し、月1回15分だけメンテする。

まず着手すべき最小セット

1. 依頼テンプレを1枚作る

目的・前提・制約・期待出力を固定

2. 複雑依頼を4フェーズ分割

調査→設計→実装→レビュー

3. 作ったカスタムプロンプト/カスタムエージェント/スキルを社内共有する

再現性を確保し、チーム共有を加速

まとめ

AI活用の成果は「使う量」ではなく「運用設計」で決まる

- 依頼設計: 伝えるより、伝わる構造を作る（良い文章より良い構造）
- CLI運用: CLI活用で高パフォーマンスの運用設計をする
- 共有資産化: 個人最適をチーム最適へ拡張する

Thanks