AIオーケストレーション時代への心構え

~~Claude Codeのサブエージェント活用の最適解を探る(仮)~~

akippa株式会社
大西 伊吹

©akippa Inc. All Rights Reserved.

オーケストレーションとは

- **複数のAIが協力して1つの大きな課題を解決する仕組み**
  - 全体を取りまとめるエージェント(メインエージェント)
  - 具体的なタスクを実行するエージェント(タスクエージェント)
  - etc
- **claude codeに、`Tasksツール`や`Agent Teams`(現在リサーチプレビュー)が登場**
  - より簡単に実施できるようになった
  - これまでは`tmux`やサブエージェントを活用して独自に実装するしかなかった

オーケストレーションの利点

- **ボトルネックになりがちな人間の判断を独立したエージェントに移譲できる**
  - 仕様書を実装に変換できる体験に近づいてきている
- **1インスタンスだと発生しがちな、作業のヌケモレが、発生しづらい**
  - メインエージェントがタスクエージェントたちの管理に専念できる
- **専門分野や立場ごとにタスクエージェントを作れる**
  - 複数の視点から、総合的な判断ができる

オーケストレーションの欠点

- **作業に途中で介入しづらい**
  - 複数AIが同時に作業するため、全てを監視して介入することは事実上難しい
- **トークン消費が大きい**
  - AI同士のコミュニケーションのためのオーバーヘッド
- **並列化しづらい作業では、利点が活かしづらい**
  - タスクエージェント1つで実行しているだけになってしまう
- **挙動の安定性は環境に左右される**
  - 全てのAIの挙動を細かく制御できないので、自然と理想的な振る舞いがされる環境を作ることが重要

AIオーケストレーションへの所感

- **数を増やすという戦略は、他の例を見ても当然の帰結に感じる**
  - CPUのマルチスレッド化、サーバーやストレージの冗長構成、etc……
  - 例えば、コンテキストウィンドウを増やしても、線形的に性能が改善するわけではない
- **AIオーケストレーションは、人の介入を減らし、自律的に行動する方向の進化**
  - 実装過程では人を介入させず、完全な成果物を最終チェックする
  - 仕様が実装に直接変換される体験(に近づいている)
- **効果的に活用するための環境整備の壁は高い**
  - 人の介入が不要な環境ができないと、トークン効率的に旨味が薄い

AIオーケストレーション時代に必要な心構え

仕様の作り込み

- 仕様を直接実装に変換できる世界が目の前まで来ている
- 人はAIよりもロングコンテキストに強い
- 人が、AIのサポートを受けて、一貫性のある仕様を設計することが重要

環境整備

- AIが迷うことなくタスクを実行できる環境を整備する必要がある
  - 怠ると、加速するAIツールの成長に追従できず、十分な恩恵が受けられない
  - アーキテクチャの破綻ない、不具合の起きづらい状態を安定させる
  - SkillやHooks、MCP、テスト、Lint、CI/CD、バージョン管理 etc……

AIオーケストレーションがあたりまえな未来に向けて

地道な取り組みをひとつひとつ積み重ねよう！

akippa

ご清聴ありがとうございました!

©akippa Inc. All Rights Reserved.