コンテキストを利用した AIコーディング

株式会社アクティヴァーチ・コンサルティング

天野 悠

1. 会社概要

会社名	株式会社アクティヴァーチ・コンサルティング
本社所在地	〒105-5539 東京都港区虎ノ門2-6-1 虎ノ門ヒルズステーションタワー39階
設立	2020年(令和2年)11月
従業員数	188名(2025年7月現在)
事業内容	**ITコンサルティング事業** **DXコンサルティング事業** **戦略/インキュベーション事業**

Activarch Consultingは、
クライアントと共に汗をかきながら課題を解決していく
ハンズオン型のコンサルティングファームです。

▶経営体制

代表取締役CEO　福井 康司

取締役COO　岩崎 洋介
（事業全般担当）

取締役CHRO　宮島 沙織
（人事・広報担当）

パートナー　田中 尚
（S&H担当）

パートナー　岩見 哲也
（S&H担当）

常勤監査役　齋藤 滋
（リスク・コンプライアンス担当）

バイブコーディングでエンジニアは不要になるか？

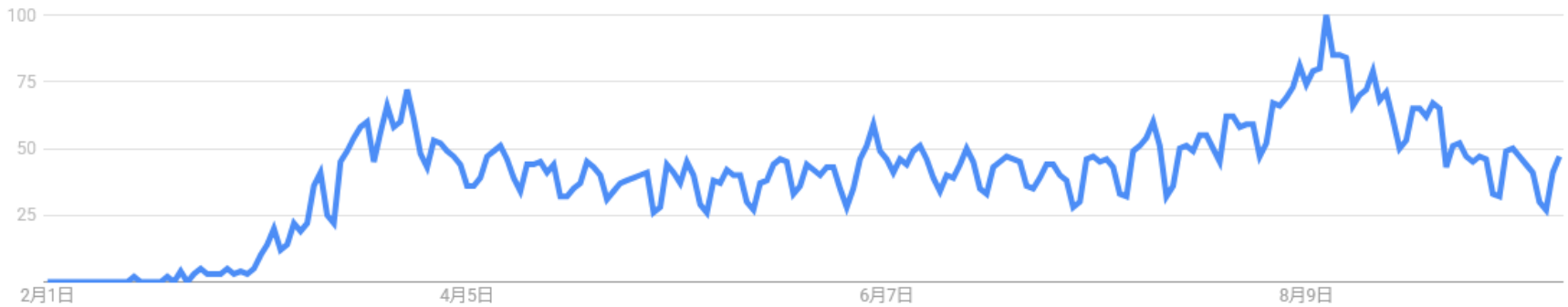
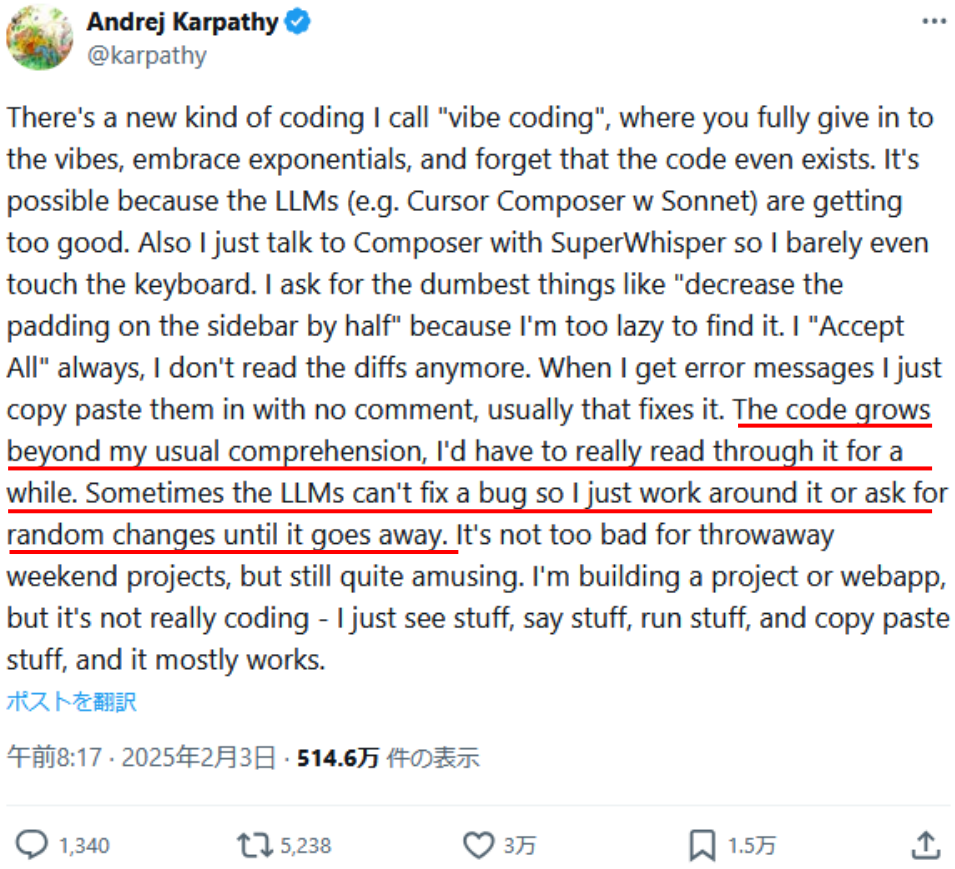


図1. Googleトレンドによるバイブコーディング検索数

2. エンジニアは必要

コーディング経験に関わらずサポートにはなるが、プロジェクトの運用・保守・拡張がしづらくなる。



Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

ポストを翻訳

午前8:17 · 2025年2月3日 · **514.6万** 件の表示

1,340　5,238　3万　1.5万

図2. Vibe coding初出
（2025/9/16時点）

理解と可観測性の欠如

“Accept All”“差分を読まない”で、コードの意図・依存関係・失敗パターンのメンタルモデルが消える。結果としてデバッグも拡張も「勘と祈り」になり、再現性・説明可能性が下がる。

品質保証の空洞化と負債の累積

エラーを丸投げ／場当たり的ワークアラウンドで進めるため、テスト・レビュー・セキュリティ検討が抜け落ちる。退行・依存膨張・性能劣化が見えないまま積み上がる。

仕様の曖昧化と設計のドリフト

根本の要件定義や設計原則が欠落し、局所最適が増える。LLM依存で意思決定が外部化され、プロダクトの方向性が徐々に逸れる。

コンテキスト伝達の必要性

3. プロジェクトの背景

社内ChatGPTについて、機能実装と環境構築, リファクタリングをする必要があった。

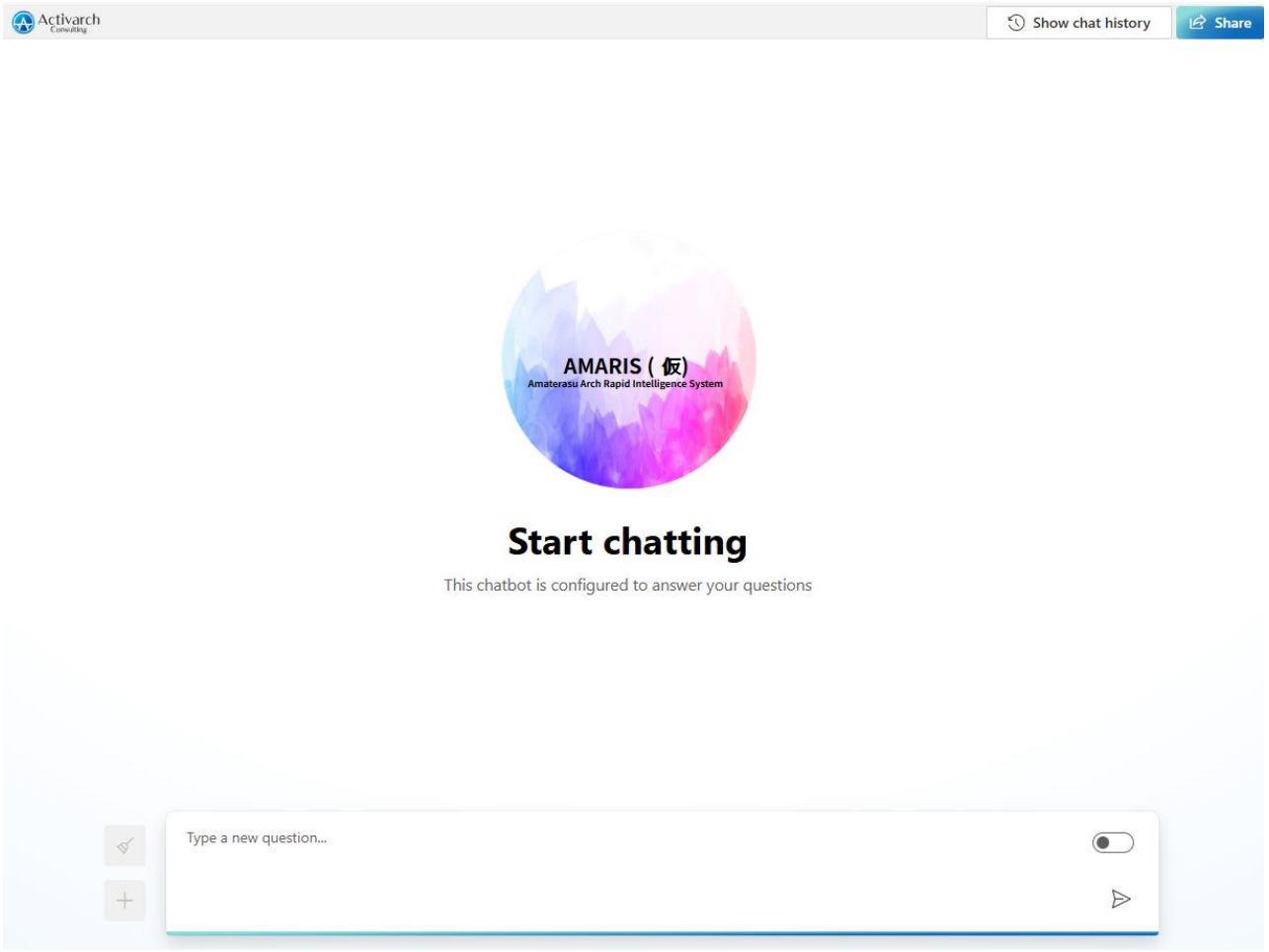


図3. 社内ChatGPT

社内ChatGPT実装
5days

Azure環境構築
3days

リファクタリング
40days～

4. 設計思想や背景情報の伝達の工夫

docsフォルダ内にリファクタリング要件やアーキテクチャ設計，実装計画，ToDo，起きた問題などを保存し、前日以前のコンテキストを入力に用いた。

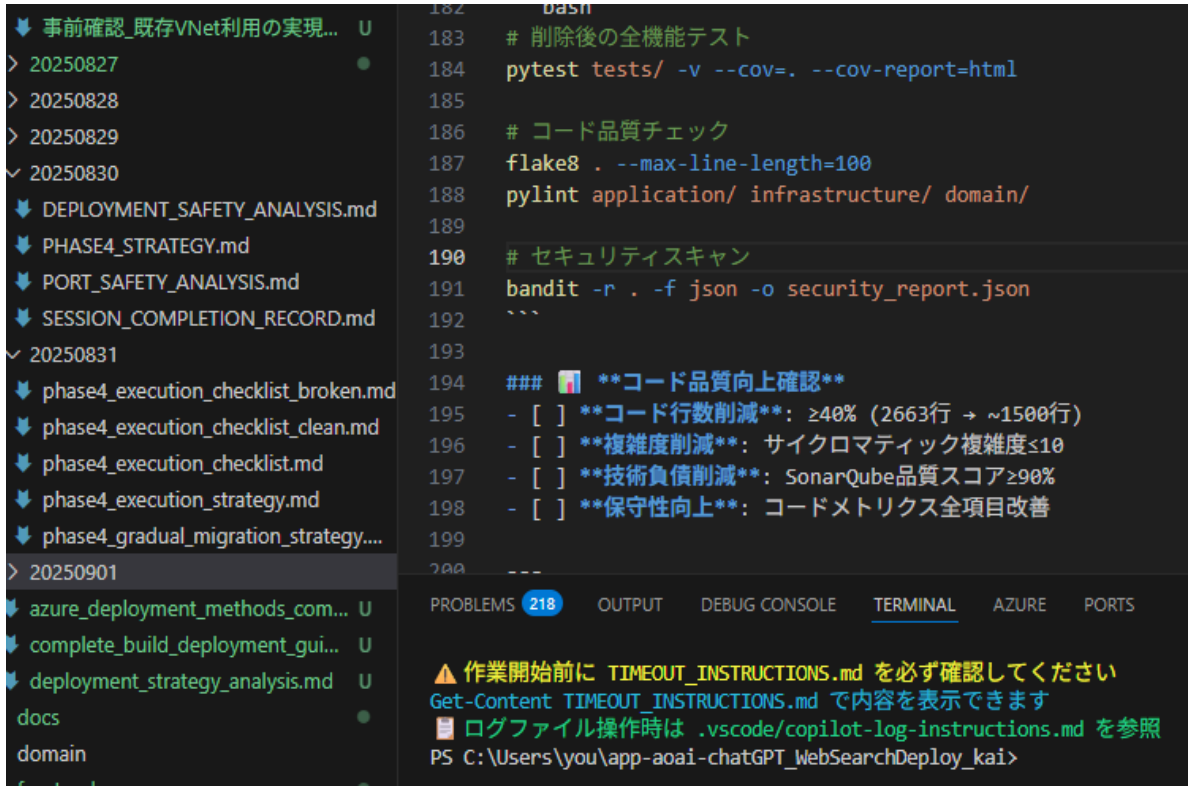


図4. プロジェクト開発中事例その１
（アクティヴァーチ社内）

5. t-wadaさんというキラーワード

テスト駆動開発原則の遵守に対し、キーワード：t-wadaさんが強力に機能した。

図5. プロジェクト開発中事例その２
（アクティヴァーチ社内）

Red（失敗するテストを書く）
まだ存在しない振る舞いを最小のテストで定義し、テストが**確実に失敗**すると確認。

Green（テストを通す最小実装）
テストを**通すためだけの最小限のコード**を書く。動けばOK、綺麗さは一旦無視。

Refactor（設計を整える）
重複排除・命名・構造化などを行うことで**内部品質を改善**する。全てのテストが**引き続き緑**であることを確認。

6. 圧倒的工数低減と今後

２ヶ月以上を見積もった工程に対し、総テスト数９００件以上を追加して８日で完了した。
今後はサービス開発に向けて開発手法を社内展開する予定である。