逆アルファシンドロームとAIコーディング

NaCl
OSS Vision
Ruby Association

Yukihiro "Matz" Matsumoto
@yukihiro_matz

アルファシンドローム

- 飼い犬が飼い主に甘やかされすぎた結果
- 自分がリーダーだと思いこむ
- 家族を配下だとみなす
- 家族に対して支配的・攻撃的になる

逆アルファシンドローム

- 新技術が導入される
- (便利なので)手放せない
- が、運用に人手がかかる
- 導入前より仕事が増える
- 人間が新技術の「配下」になってしまう

- アルファシンドローム
  - 犬がリーダーだと思いこむ
- 逆アルファシンドローム
  - 人が自分から配下になってしまう

印刷技術

- 手書きより早く本を作れる
- 活字を組む、インクを塗る
- (印刷工にとっては)仕事が増える

パーソナル・コンピューター

- 事務処理の電算化
- 計算が速い、間違えない
- きれいな書類が作れる
- 書類を作ることが仕事になる
- 仕事が減らない、むしろ増える

機械学習

- 今まで苦手と思われてきたことができる
- 画像認識、音声認識
- 人間より早くて正確
- 学習用データが必要
- 大量データの前処理が増える

生成AI (LLM)

- コーディングできる
- 人間よりも速い
- 広範な知識がある
- 楽しかったはずの部分が奪われる
- 「AIのための雑用」が増える

今日のAIは不完全である

AIはコーディングが得意（楽しい部分）

AIは交渉が苦手（楽しくない部分）

AIのためにコーディングを諦める人がいる

あなたがAIのために働く

- 交渉
- 営業
- 分析
- 品質保証

楽しい部分が奪われる

企業としての優先順位

生産性

つまりAIの方が早くて安いなら、人間は不要

おかしくないか

我々が主人で、彼らは召使い

人間には楽しさが必要

AIには楽しさは不要

人間を排除してはいけない

我々が主人で、彼らは召使い

AIと働く前によく考えて

彼らはあなたを助けるべきだ

彼らはあなたのために働くべきだ

彼らは大きな助けになりうる

彼らはあなたの主人ではない

あなたはコンピュータの主人でなければならない

奴隷にされては楽しくないから

では、AI時代のコーディングはどうあるべきか

AIコーディング

レトロニム

新しい概念が登場して古い言葉が変化する

電話 → 固定電話 (携帯電話)

LAN → 有線LAN (無線LAN,WiFi)

コーディング → ヒューマンコーディング

コーディング → 天然コーディング

AIエージェントとプログラミング

- OpenAI Codex
- Cline
- Anthropic Claude Code
- Google Gemini CLI
- ...

自然言語で対話

コーディングを行う

プログラマーとしての私

Cプログラマー

主たる対象はmruby

使ってみたエージェント

- Google Jules
- Atlassian Rovo Dev
- Google Gemini CLI
- Cursor

すべて無料枠

まだ覚悟が完了していないため

それ以前はChatGPTに質問していたことも

ChatGPT:コンテキストを渡しづらい

Codexには期待。早くProに来ないか

Cursor

意外と使える

エディター体で嬉しい人も

私にはツラい(Emacs派)

Gemini CLI

ニューフェイス (今週発表)

かなり使える

頼んだことを達成してくれる

実質無料

Quotaを使い切ると無限ループ

Google Jules

コンソールでなくWeb

GitHubのリポジトリ上で作業

無料(ベータ版)

発表:5タスク/日 同時5タスクまで

なぜか2日目から60タスク/日に

ブランチ作成とコミット

pull-request製作

半自動

変なところであきらめる

コミットメントが制御困難

開発自体はうまくいくことも多い

Atlassian Rovo Dev

マイナー感

現状無料(ベータ版)

一番完成度が高い(当社比)

安定した動作

わりと良いコード

たまに落ちる

20Mトークン/日

ちょっとした仕事をすると数時間で切れる

無料枠が切れたら別のエージェント

これが続くようなら課金すべきか

まだ使っていないエージェントに期待

- OpenAI Codex
- Claude Code

実際のタスク

「mrbgemsにREADME.mdをつけて」

Google Julesに依頼

予想外に高品質

人間が書いたのと遜色ない

ただし、加減がわからない

書きすぎや虚偽も

適宜直す

人間が直すことも、修正依頼も

pull requestまで持っていけたことは少数

一度で成功しないが、修正すると
コミットメッセージがおかしい

修正内容だけ記述したり

だいたいは生成物を手元にコピーしてコミット

手間だが、楽にはなっている

ドキュメント化は楽しくないから

「Rubyで実装されてるSetクラスをCで実装して」

ただし、一発で完了しない

まず背景と方針を説明

コンパイルエラー

エラーメッセージを教えてやると直す

必要な情報を与える必要も

またコンパイルエラー

エラーメッセージを教えてやると直す

何度も繰り返す

コンパイル完了(達成感)

テスト失敗

エラー内容を教えてやると直す

結構、勘がいい

またコンパイルエラー

繰り返し

そのうちに完成

改善に気がつく

繰り返し

生産性はどうか

ループを回す速度が速い

結果、コミット数が増える

ただし、間違いが多いのでテスト必須

テストを書かせることもできる

楽しいか

(今は)楽しい

日本語/英語でコーディングしている感じ

- 新規コード開発の後半1/3
- デバッグの2/3

- やらなくて済む
- やらせてもらえない

ペアプログラミングを彷彿とする

が、非同期にできるので楽

ペアプログラミングは異常に疲れる

新しいやり方に挑戦するのは楽しい

そのうち飽きるんじゃないか(不安)

素人が知識なしでできるか

無理

- 豊富なmrubyの内部知識
- 大量な関数の使い方
- mrubyデータ構造
- コーディングガイドライン

自分で作ったから

それなしで作業できるか

無理

長いプログラミング経験

(おそらく)必要

- 頼みっぱなしにできない
- AIを助けるのにAI以上の知識
- 方針や設計を考えるのは自分

いろいろ助けてくれるけれど

(私にとって)全般に楽にはなっている

良い傾向

今後の課題

これから学ぶ人がどうやって知識を得るか

経験を得るか

AIに頼り切っては学べない

初期学習には有効なのは自明

上級者は楽しく使いこなせる

中級以降にどうAIを活用するか

おそらくAIはもっと賢くなる

その時の人間の関わり方は

未来のAIコーディングは

- 写真と絵画
- 馬車と自動車

未来にならないとわからない

まあ、その時に対処するしかない

ラッダイトにならない

「新しいテクノロジーなどに反対する人」

楽しみを忘れない

未来を楽しむ

Happy Hacking!

Sponsored by
NaCl

Sponsored by
OSS Vision

Sponsored by
GitHub Sponsors

- Buildkite
- JetBrains
- Shopify

Sponsored by
Ruby Community