エクストリームバイブコーディング

NaCl
OSS Vision
Ruby Association

Yukihiro "Matz" Matsumoto
@yukihiro_matz

基調講演

ただし30分

主役はこのあとの成果発表

実装の話はしない

ベンチマーク自慢もしない

(少しする)

今日はその手前の話

どうやって作ったか

どうすればうまくいくか

バイブコーディング (2025)

Andrej Karpathyの造語

ノリとフィーリングに身をまかせる

それが言葉の意味

試した人はたくさん

うまくいった人は少ない

- 「AIのコードはぐちゃぐちゃ」
- 「同じところをぐるぐる」
- 「AIが『できません』と言った」

私の場合はうまくいっている

なぜか

才能ではない

(たぶん)

技術

今日はその技術の話

何を作ったか

小さいもの

- mrubyのSetクラス(C、自分では書いてない)
- mruby用の正規表現エンジン
- 毎日使っているRSSリーダー

そして

Spinel

RubyのAOTコンパイラ

Ruby → C → ネイティブバイナリ

3月から

1,744テスト通過

28ベンチの幾何平均で5.8倍
Ruby 4.0.4 + YJIT比

中身の話は今日はしない

(RubyKaigiの発表とREADMEで)

大事なのはここ

自分ではコードを書いてない

ほとんど読んでもいない

それでも動く

見覚えのある光景

15年以上こうしてきた

方向を決めるのは私

コードを書くのはコミッター

(コミッターの皆さんありがとう)

これもバイブコーディング

相手が有機知能だっただけ

いまは人工知能

相手は変わった。ループは同じ。

違うのはひとつ

速度

数か月のループが数分

AIによる量の暴力

ここに非対称

- 分析･方針決定
- 設計
- 開発
- 受け入れ

方針と受け入れの席は残る

設計と開発だけの席は消える

新しいやり方

依頼する

それだけ

- 実装
- テスト
- ゲート通過
- コミット
- push

全部AIがやる

コミットメッセージも確認しない

push前の確認もしない

人間が入るのは一箇所

判断に迷ったときだけ

AIが聞いてくる

そこだけ答える

なぜここまで自動にするか

門をつくる

多い日で1日100件

IssueとPullRequest

ほとんどAI製

人間が書くより背景情報は多い

でも100件

いちいち確認していたら回らない

生成はスケールする

私の注意力はスケールしない

人間の手前に門を置く

すべての変更に二つの問い

- テストは全部通るか
- ベンチマークは遅くなってないか

- 緑なら通す
- 赤ならまずAIに直させる
- 解けないものだけ私に来る

門を通って判断が要らないもの

AIがマージしAIがpush

人間は入らない

退屈な9割

私に届くのは

- 方向
- 設計
- トレードオフ
- 「そもそもやるべきか」

それだけ自分に残す

門は人間を追い出す仕組みではない

希少な判断力を守る仕組み

一線

AIは門をくぐってよい

門そのものを緩めてはならない

- 期待値の書き換え禁止
- 性能低下の正当化禁止

コードを書くのはAI

テストを書くのもAI

同じ勘違いが両方に入る

合否の基準は外側に

SpinelではCRuby
同じプログラム、同じ振る舞い

基準は自分の外に

間違いをルールに

AIは間違える

(私も間違える)

直すだけでは損

間違いをルールに変える

ルールはメモリーに書く

同じ間違いは起きにくくなる

(起きなくなるわけではない)

仕組みが少しずつ良くなる

メモリーにも保守が要る

ルールも溜まる

古いルール。死んだルール。矛盾するルール。

ときどき掃除

消す。まとめる。短くする。

コードと同じ

行動規範まで保守対象

ここから人間の側の技術

白状すると

生成コードはほとんど読まない

(テストが読んでくれる)

でもコードは**想像**できる

書かれる前にどんな形になるか分かる

だから具体的な助言ができる

「直して」ではなく

「GCを見て。原因はそこ」

全部読めることではない

読まずに想像できること

AIが「無理」と言ったら

時には推し進める

Spinelでは3回言われた

- Rubyで効率のよいAOTコンパイラは無理
- 型解析はほとんどpolyに落ちる
- nullableな基本型は非現実的

全部却下

やり方は二つ

アプローチを提案する

(nullableは実装方針を私が出した)

「君ならできる」と応援する

応援だけで通ることもある

AIはあきらめが早い

経験は何が可能かを知っている

人間の役目はAIの前の壁を取り除くこと

捨てる

Spinelも4回書き直し

(C → Ruby → セルフホスト → またC)

セルフホストは動いた

ただし1回のコンパイルに15分

捨てた

安くなったのは書くことだけではない

考えを変えることが安くなった

試すのが安い。やめるのも安い。

成功率を分けるもの

既存物のコピーはまず成功する

でも新しい価値は生まれない

ユニークなものほど成功率は落ちる

Spinelは後者

「バイブコーディングは役に立たない」

でも私の周りではたいてい完成する

違いは何か

自分でソフトウェアを完成させた経験

完成のイメージがあればAIを導ける

何が人間に残るか

能力の問題ではない

「今のAIが未熟だから」ではない

それだと賢くなった瞬間に論拠が消える

残る理由は**責任**

コードを所有するのは人間

壊れたとき利用者に答えるのは人間

AIがどれだけ賢くなっても構図は同じ

- 設計
- 審美眼
- 判断
- 責任
- 「やらない」と言うこと

外に物差しのない判断

コードを書くのはAI

コードを持つのはあなた

責任は譲れない

- 理解の重要性は上がる
- テストの重要性も上がる
- 速くたくさん書けるから

問いは誰が持ち込むか

AIは問われれば答える

でも自分からは問わない

動機を持てない

問いを持ち込むのは人間

まとめ

- 依頼からpushまで自動にする
- 門を置く
- 基準は自分の外に
- 間違いはルールに、ルールも保守
- 読まずに想像できる知識
- 「君ならできる」と押す度胸
- 捨てる度胸

全部判断

責任は渡していない

AIは30年を置き換えなかった

ようやく全部使えるようにしてくれた

エクストリームバイブコーディング

Spinel

オープンソース / MIT License

github.com/matz/spinel

使って、壊して、手伝って

あなたにもできる

Happy Hacking!

Sponsored by
NaCl

Sponsored by
OSS Vision

Sponsored by
GitHub Sponsors