この1年のコーディングエージェントの進化と今後半年の予想

# プログラマーAGIは、もう来ていた

コーディングエージェントの1年と、次の半年



みんなのPython勉強会 in 長野　2026-07-19
さとうきよし（@stealthinu）

# 自己紹介

## さとう きよし （@stealthinu）

- 白馬村在住でジーワークスという小さな会社をやってます

| | |
|---|---|
| 1990年代前半 | ネオコグニトロンで文字認識をする研究室で眼球運動を測定していた |
| 2022年から1年半くらい | VRメタバース内での音声変換（VC）とニューラル圧縮 |
| 2023年半ばから | **LLMを利用したPJマネジメントツールの研究開発　いまはAI生成コードの品質保証・検証まわりが中心** |

- 以前は迷惑メール対策とかよくやってました

# このシリーズの3回目です

2025-04
**MCPとエージェント**
LLMが外の世界へ手を伸ばす仕組みが整った

2025-07
**「現在地」だけでなく「速度」も見よう**
限界そのものより、限界が動く速度を見るべきだ

2026-07
**今回**
その1年後、コーディング領域では人間水準という閾値を越えたのではないか

1年前の発表では「半年後に大規模開発が可能になる」と予想した。
今回はその**答え合わせ**であると同時に、**次の半年**を考える続編

今日の結論を先に

# プログラマーAGIは、<br>もう来ていた

プログラマーの仕事がすべて自動化されたわけではない。
だが「与えられた目的に対してコードを調査・実装・テスト・修正する知能」は、
平均的な人間プログラマーの水準に到達し、すでに超え始めている

# 今回の「プログラマーAGI」の定義

### 含める能力

- 未知・大規模なコードベースを探索する
- 要求から変更箇所を特定する
- 複数ファイルをまたいで実装する
- CLI・コンパイラ・テスト・ブラウザ等の道具を使う
- 失敗結果を読み、修正して再実行する
- リファクタ・移植・バグ修正を長時間継続する
- 必要に応じて計画を変え、成果物まで持っていく

### 含めない・別の論点にする能力

- 顧客自身もまだ知らない要求の発見
- 何を作るべきかという経営・製品判断
- 法的・社会的責任を負うこと
- 長期的な保守方針や組織内の利害調整
- すべての出力を無条件に信用できること

境界を絞るのは主張を弱めるためではない。**「コーディングという狭いが巨大な領域では、すでに閾値を超えた」**を明確にするため

# 軸1　モデルとエージェントの進化

2025年前半は「エージェントらしく動く」こと自体がニュースだった
2026年夏には、長時間・大規模・複数エージェントが前提になった

# 2025年　エージェント元年から競争へ

| | | |
|---|---|---|
| 2024-11 | ● | MCP公開（Anthropic）　エージェントと外部ツールをつなぐ標準化 |
| 2025-02 | ● | **「Vibe Coding」命名（Karpathy）／「CLINEに全部賭けろ」（mizchi）／Claude Code研究プレビュー** |
| 2025-05 | ● | **Claude 4＋Claude Code一般提供、Codex、Julesが出揃う** |
| 2025-07〜09 | ● | Kiro、GitHub Spec Kit、GPT-5、GPT-5-Codex |
| 2025-11〜12 | ● | Claude Opus 4.5、GPT-5.2 / 5.2-Codex　Claude対Codexの競争が本格化 |

**発表者の実感**　2025年5月、Claude Codeが「実務で使う道具」になった。当時のCodexはクラウドサンドボックスの非同期実行で、ローカルを直接触るClaude Codeの方が日常開発に入りやすかった。2025年後半からCodexが急速に追い上げた

# 2026年　分水嶺を越えた

| | | |
|---|---|---|
| 2026-02 | ● | **Claude Opus 4.6 / GPT-5.3-Codex**　ここで一段階変わった |
| 2026-03〜04 | ● | GPT-5.4（1Mコンテキスト、ネイティブなPC操作）、GPT-5.5、Sonnet 4.6（数か月前のOpus級を安価に） |
| 2026-05〜07 | ● | **Opus 4.8、Fable 5、Sonnet 5、GPT-5.6 Sol / Terra / Luna** |

### 2026年初頭

「平均的な人間プログラマーと実装速度・完成力で競争する段階」を越えたように見える分水嶺

### 2026年夏

Fable 5やGPT-5.6 Solでは、実務感覚として「コーディングはAIの方が上」と表現した方が近い

# 軸2　開発作法の変化

バズワードが変わったのではない。ボトルネックが変わった

# 「一番困る場所」が移ってきた

**Prompt Engineering**
どう言うか

→

**Context Engineering**
何を見せるか

→

**Spec / Harness Engineering**
どんな環境と制約で働かせるか

→

**Loop Engineering**
どう自動で繰り返し、検証し、止めるか

| | |
|---|---|
| **Prompt時代** | ボトルネックは「うまく命令できないこと」 |
| **Context時代** | 「必要な情報がモデルから見えないこと」（2025-09 Anthropicが公式記事） |
| **Harness / Loop時代** | 「環境・検証・繰り返しの設計」（2026-02 OpenAI Harness Engineering、2026-06 Loop Engineering） |

年代は厳密に分離しない。モデルの能力向上で「一番困る場所」が移った図として見る

# Spec-Driven Developmentと、その先の問題

### SDDの登場

- 2025-07 Kiro（AWS）、2025-09 Spec Kit（GitHub）
- Vibe Codingへの反動として、仕様を先に構造化し、計画・タスク・実装へ落とす
- モデルが仕様を実装できる段階に来たため、「正しい仕様を渡すこと」が中心課題になった

### ただし、限界がある

問題は、仕様を正確に文章化できないことだけではない。

**顧客やユーザー自身が、動くものを見るまで欲しい仕様を知らないことがある**

その意味で、Vibe Codingは雑な開発ではなく**「高速な仕様探索」**として再評価される可能性がある　→　後半の未来予想へつながる

# Human-in-the-loop から Human-on-the-loop へ

### Human-in-the-loop

- AIが途中で止まり、人間の承認や修正を待つ
- Claude Codeでは許可プロンプトの**93%が承認**され、承認疲れが問題になったとの報告
- 人間が毎回承認する方が安全、とは限らない

### Human-on-the-loop

- AIのループは自走し、人間は外から監視。異常時や判断時だけ介入する
- 経験者ほど自動承認を増やす一方、必要時には中断もする（Anthropic 2026-02調査）
- 「放置」ではなく、監視の粒度が上がる

**人間がループに入ると品質が上がった時代**から、**人間がループに入ること自体が速度のボトルネックになる時代**へ

# 軸3　接続・環境・制御の変化

周辺ツールの流行は、その時点のモデルが何を苦手としていたかの履歴である

# MCPからSkillsへ、ではなく役割分担へ

### MCP 「接続」の標準

- 2024-11公開。モデルが外部データ・ツールから孤立していた時代の答え
- 2025年前半、対応が広がり大きな話題に
- 2025-12、Linux Foundation傘下AAIFへ寄贈
- 「終わった」のではなく、**業界標準インフラへ沈み込んだ**

### Skills 「手順と知識」

- 2025-10公開、2025-12にオープン標準化
- 指示・スクリプト・資料をフォルダに置き、必要な時だけ読む単純な仕組み
- モデルがCLI・README・手順書を扱えるようになったから成立
- 高性能モデルには、常時スキーマを見せるより**必要時に手順書を読む方が効率的**な場合がある

話題の中心が移ったのは、**モデルの弱点とボトルネックが変わったから**。MCP回で話した内容は、いまはインフラとして下の層にある

# ハーネスは、AIを縛るものから、AIを働かせるものへ

### 以前　檻としてのハーネス

- 変なことをしないように檻へ入れる
- 1操作ごとに承認する
- 人間が常時付き添う
- クラウドサンドボックスで隔離実行（初期Codex、Jules）

### 現在　働く環境としてのハーネス

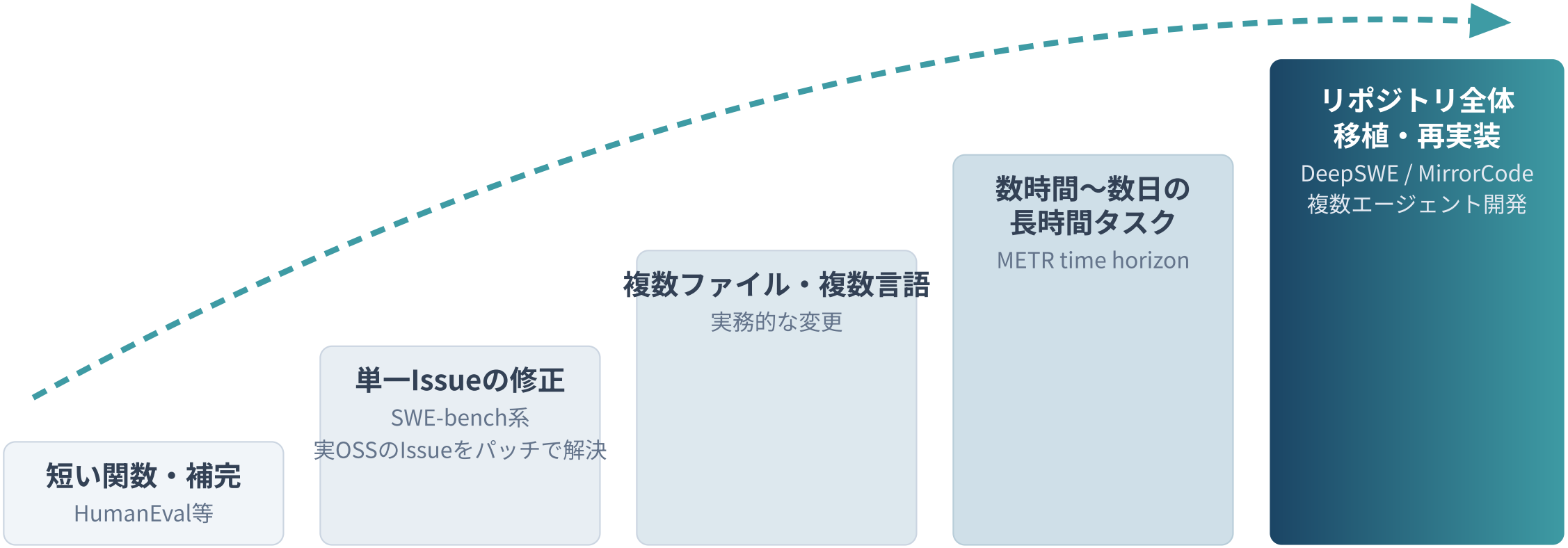
- 長時間仕事を続けるための記憶と進捗管理
- テスト・レビュー・コンパイラを自動のフィードバックにする
- 複数エージェントを並列に動かす
- 異常時だけ人間へエスカレーションする

「ガードレールが不要になった」ではない。モデルが信頼できるようになったため、制御の目的が**「暴走を止める」**だけでなく**「大きな仕事を安全に任せる」**へ変わった。権限の「被害半径」を制御する必要は残る

# 軸4　評価尺度と作業規模の変化

ベンチマークが変わったのは測定技術の流行ではない
以前は現実的でなかった規模の仕事が、測れる対象になったからである

# ベンチマークが測る「仕事の大きさ」




**ベンチマークの難易度は、昨日まで人間の仕事だった範囲の地図である**

# 新世代ベンチマークと、時間軸の物差し

**DeepSWE** 2026-05

- 91のアクティブOSS、5言語
- 113の新規長時間タスク
- 短く自然な要求から探索・実装させる
- 参照解は平均668行追加・7ファイル編集
- 外部挙動を手書きverifierで検証

**MirrorCode** 2026-06

- ソースを与えず、入出力挙動の観察からプログラム全体を再実装させる
- 人間でも十分な技術力と時間が要る仕事が、評価項目になったこと自体が象徴的

**METR time horizon**

- 人間の専門家が要する時間で仕事の大きさを測る
- 50%成功のtime horizonは**約7か月ごとに倍増**と報告

**対照例**　METRの2025年のRCTでは、当時のAIツールで熟練OSS開発者が本人の予想に反し平均**19%遅くなった**。その後METR自身が「急速な進歩により現在のツールを代表しない」と追記。**1年前のスナップショットは、それほど短期間に古くなる**

# 大規模開発　半年で実験になり、1年で実用になった

### 半年前　大規模「実験」

**2026-01　Cursor　ブラウザをゼロから**

数百エージェント並列、約1週間、100万行超・1000ファイル

**2026-02　Anthropic　Rust製Cコンパイラ**

16のClaude、約2000セッション、約2万ドル、約10万行。Linux 6.9をx86 / ARM / RISC-V向けにビルド可能

### 今　実運用への「実用」

**2026-06　Stripe　5000万行Rubyの全体移行**

チームなら2か月超の移行を1日で（Anthropic発表によれば。第三者検証は限定的）

**2026-07　Bun　Zig→Rust全面移植**

実プロダクトのランタイムを移植し、本番投入　→　次のスライドで詳細

大規模コードが生成されたことと、製品品質で完成したことは同義ではない。**実験と製品事例を分けて見る**

# BunのZig→Rust全面移植　2026-07-08公開

**53.5万行**
もとのZigコード（コメント除く）

**11日間**
1人のエンジニア＋Fable 5＋約50のDynamic Workflows

**6502**
コミット数

- すべてのプラットフォームでテスト**100%通過**、実運用へ投入。既知の移植起因回帰は公開時点で19件・修正済み
- Claude Code自身も、Rust版Bunを本番利用している

半年前には「大規模なものを作る実験」だった。1年後には、**何千万回もダウンロードされるランタイムの実移植**に使われた

# 4本の証拠が、同じ結論を指している

1

**モデル／製品**

長時間・大規模コード・ツール利用・自律的な修正が標準機能になった

2

**開発作法**

人間が一手ずつ指示する前提から、ループを設計して監視する前提へ

3

**周辺技術**

檻と許可プロンプトから、長時間働かせるハーネスへ

4

**評価／実例**

Issue修正から、リポジトリ規模・移植・再実装・数日単位の仕事へ

別々の方向から見ても、同じ変化を示している

# 実装勝負は終わった

人間の主戦場は、コードを書くことから
何を作るか・どう確かめるかへ移る

- ソフトウェア開発の全工程が終わった、という意味ではない
- 人間の仕事が明日消える、という意味でもない
- ただし**「実装は人間の希少能力」という前提は、もう置けない**

# 今後半年の予想

技術側とプロセス側の、2本立てで考える

# 予想①　最強が普通になる

重要なのは最高性能の更新だけではない。**今の最高性能が、より安く・速く・日常的に使える層へ降りてくる**

### すでに起きている根拠

- 2025年末のOpus級能力が、2026-02のSonnet 4.6価格帯へ
- Sonnet 5は数か月前のOpus 4.8級を低価格で
- GPT-5.6は「1トークン・1ドルあたりの有用な仕事」を前面に
- Qwen3.6-27B等、小型公開モデルも1年前の高性能帯へ
- METRの尺度は約7か月で倍増という傾向

### 技術者に来る変化

- 今は高くて常用できないモデルを、毎日の雑な作業にも使える
- 実装・レビュー・テスト・調査に複数体を常時並走させる
- 限界が人間のタイピング速度ではなく、計算資源と検証能力になる
- 数万〜数十万行の生成・移植が通常の選択肢になる

半年後には、今の「高級な知能」を燃やすことへの心理的な抵抗が消え始める

# 予想②　新しいボトルネックが前へ出る

## 実装が制約でなくなると、意思決定と検証が制約になる

1

### 仕様発見

顧客は動くものを見て初めて「欲しいもの」が分かる。生成→操作→反応→仕様更新の高速ループが必要になる

2

### 検証と品質保証

人間が全diffを読む方式はスケールしない。テスト・型・静的解析・複数モデルレビューで**機械的証拠**を積み上げる

3

### ループを閉じること

失敗を分類し、修正し、再実行する。いつ停止し、いつ人間へ上げるか。**ループの設計品質が出力品質を決める**

4

### システム理解と意図の説明

なぜこの設計か説明できるか。何を保証し何を保証しないか。事故時に原因を追跡できるか

自己点検　発表者自身が検証・品質保証を研究しており職業上のバイアスの可能性はある。ただし各社の公開実装も、モデルの賢さそのものより検証可能なループと環境へ重心を移している

# おまけ　この資料自体も、1年前とは違う作り方をしている

| | |
|---|---|
| 約2年半前 | LLMはタイトル候補や短い文章のたたき台に使える程度 |
| 1年前 | Claudeと相談しながら構成を作り、スライド本文も生成 |
| 今回 | **過去資料を読ませ、リアルタイム音声で長時間壁打ちして設計書を作成。スライドはコーディングエージェントが生成** |

音声会話は「予想より早く来た」ではなく「やっとまともな全二重会話が来た」が正直な感覚。重要なのは音声の新規性ではなく、**会話相手が論点を保持し、構成候補を作り、事実確認へつなげ、資料の設計図まで作れるようになったこと**

# 前回予想の答え合わせと、次の1年

| 1年前の予想 | 半年後 | 1年後（いま） |
| --- | --- | --- |
| 大規模プロジェクトを自動開発 | Cコンパイラ・ブラウザ等の大規模実験 | **Bunの実用コードベース全面移植** |
| 人間はAIの環境を整備する側へ | Harness Engineeringが言語化 | **Loop Engineering、Human-on-the-loopへ** |
| GUIを含む作業の自動化 | ブラウザ操作・Computer Useが一般化 | **GPT-5.4以降はネイティブなPC操作を統合** |

前回「僕の仕事はあと2年半でなくなる」と言った。1年たって、なくなったというより**実装から検証・仕様・ループ設計へ仕事が移動した**。では次の1年で、その仕事もどこまで自動化されるか

# 人間を超えたか、ではない

**どこまで離れるか**
**どこまで安く、普通になるか**

半年後の「普通」を、いまの上限から想像しない

# 参照

**この資料一式・発表設計資料**

github.com/stealthinu/talks

**構成を検討したChatGPT（gpt-live）との壁打ちログ**

chatgpt.com/share/6a5ae50c-757c-83e8-81ca-740da74170f6

**過去2回の発表**

MCPとエージェント（2025-04） speakerdeck.com/stealthinu/mcptoeziento

生成AIの「現在地」だけでなく「速度」も見よう（2025-07） speakerdeck.com/stealthinu/...

# ありがとうございました

ご質問どうぞ

さとうきよし（@stealthinu）

みんなのPython勉強会 in 長野　2026-07-19