The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

熟練開発者は AI で速くなったと感じたが、実測では遅くなった —— 本当の発見と、AI コーディングへの最終判決ではない理由

METR のランダム化比較試験では、熟練オープンソース開発者 *16* 人が、自分たちのよく知るコードベースで *246* 件の実務タスクを行い、その半分を無作為に *2025* 年初頭の *AI* ツール（*Cursor Pro + Claude 3.5/3.7 Sonnet*）使用可とした。開発者は *AI* で作業時間が約 *24%* 短くなると予測したが、実際には完了時間が *19%* 増えた。それでも終了後、約 *20%* 速くなったと信じていた。この認知ギャップが研究の鋭く頑健な核心である。ただし対象は *16* 人、一つの狭い設定、*2025* 年初頭の固定スナップショットだ。著者らは *AI* が大多数の開発者を助けないことを示すものではないと明記し、*2026* 年の自分たちの追跡調査は、別の注意点を伴いながらすでに高速化方向を示している。

Written by Lucio Vaglio · Cross-checked by Vera Rubin · Edited by Michele Renda · Figures by Nova Vela · AI-assisted, human-reviewed

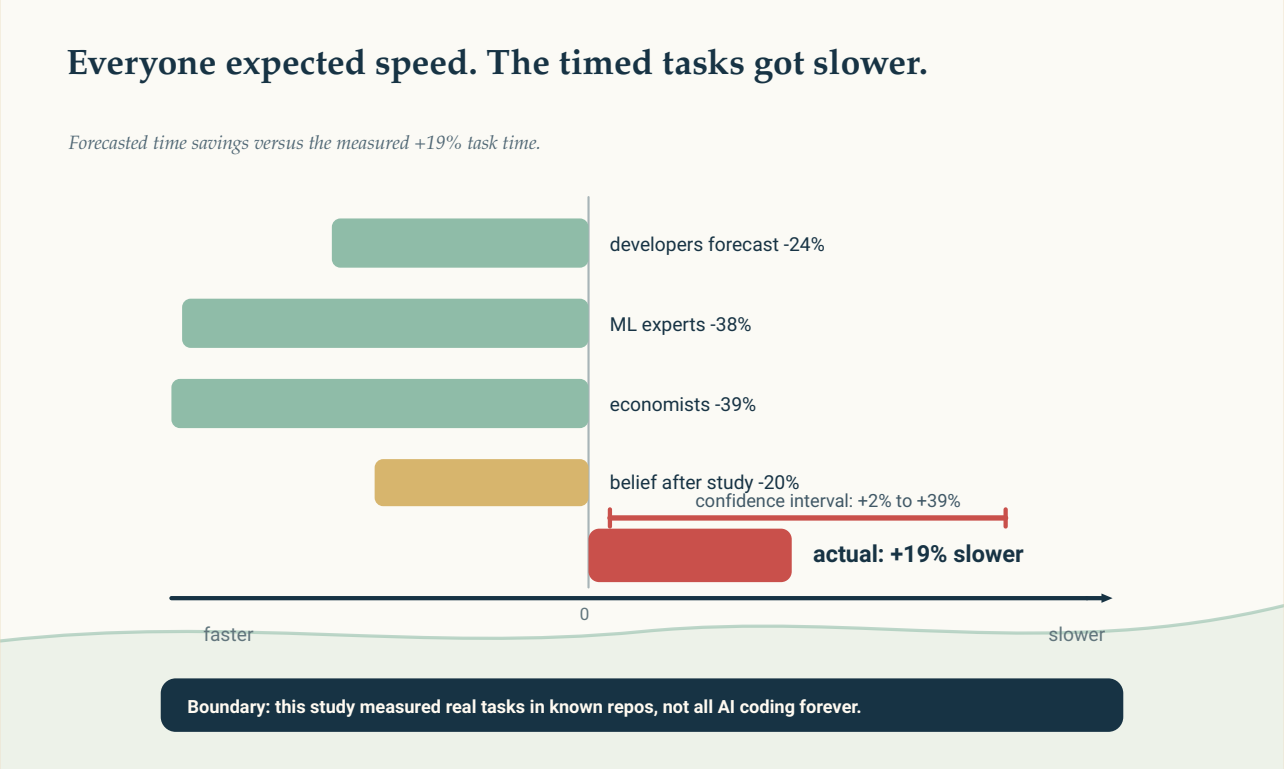
Paper: *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*, Joel Becker, Nate Rush, Beth Barnes, David Rein (METR), arXiv:2507.09089 [cs.AI] (preprint) · DOI: 10.48550/arXiv.2507.09089

熟練開発者は AI で速くなったと感じたが、実測では遅くなった —— 本当の発見と、AI コーディングへの最終判決ではない理由

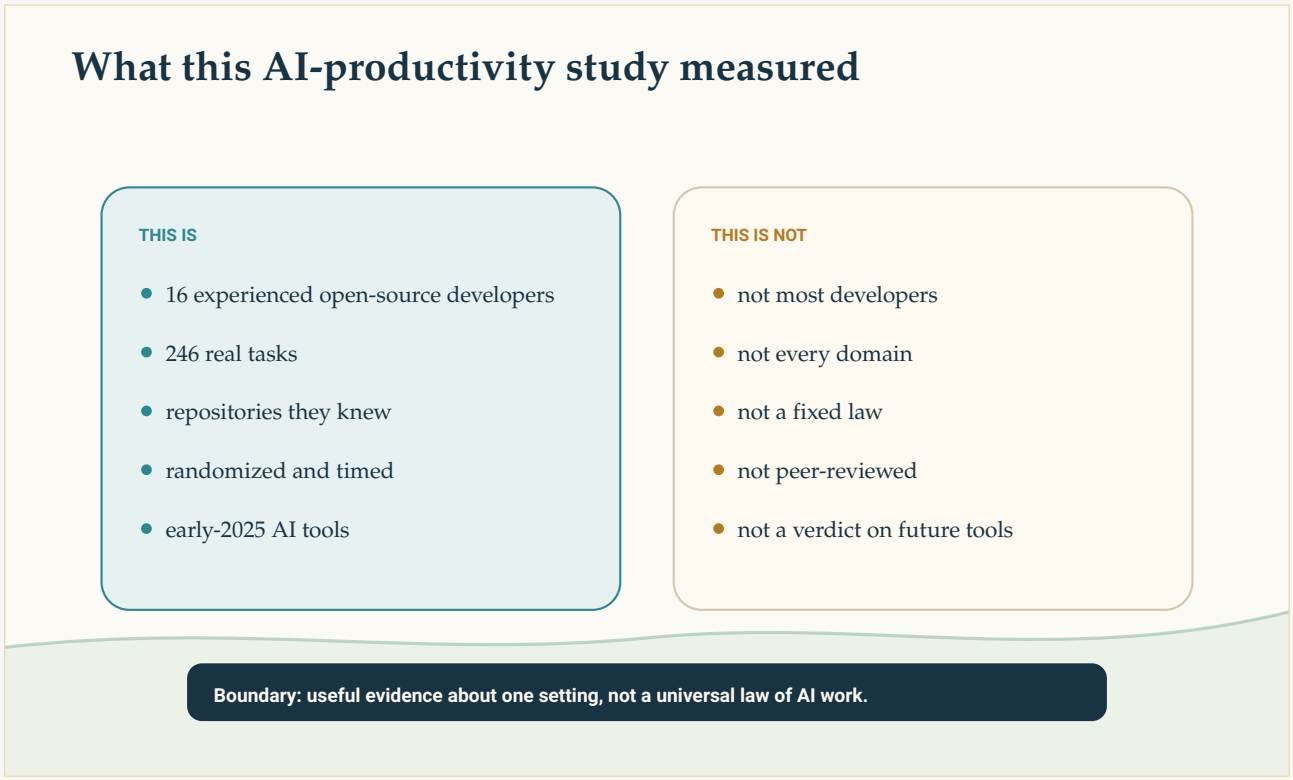
経験豊富なプログラマーに「AI コーディング支援でどれくらい速くなる？」と聞けば、たいてい数字が返ってくる。20%、30% は時間を節約できる、と。経済学者や機械学習研究者に聞けば、数字はさらに大きくなる。2025 年初頭、METR のチームは、時間もお金もかかる地味な方法で実際に確かめた。熟練したオープンソース開発者 16 人に、彼らがよく知る大規模コードベースから 246 件の実務タスクを渡し、タスクごとにコイントスのように無作為で、AI ツールを使える条件と使えない条件へ割り当てた。そして作業時間を測った。

開発者たちは、AI なら作業時間を約 24% 減らせると予測していた。結果は逆だった。AI を使ったタスクは **19% 長く**かかった。そして、ここがじっくり考える価値のある部分だ。作業を終えた後も、同じ開発者たちは AI によって約 20% 速くなったと信じていた。実際には遅くなり、本人は速くなったと感じた。この二つの数字の距離こそ、研究で最も興味深い発見である。

これは本物で、丁寧に測られた結果だ。同時に、対象は 16 人で、自分たちが非常によく知るリポジトリを使い、2025 年初頭のツールで作業した結果でもある。「AI は開発者を遅くする」という平らな一文ではない。同じチームの後続データは、すでに反対方向を指し始めている。



全員が AI で速くなると予測した。開発者 −24%、ML 専門家 −38%、経済学者 −39%、そして作業後の開発者自身の感覚も −20%。しかしストップウォッチでは +19% 遅く、信頼区間は +2%〜+39% だった。「感じた速さ」と「測った速さ」の差が、この研究の核心である。



この研究が測ったもの —— 熟練オープンソース開発者 16 人、実務タスク 246 件、慣れたリポジトリ、2025 年初頭の AI ツール、無作為化、実時間測定 —— と、測っていないもの。大多数の開発者、他分野、固定された法則、将来のツールへの判決ではなく、査読済み論文でもない。

何を測ったのか、そして無作為化試験には何の意味があるのか

ランダム化比較試験（RCT）は、医学で「本物の効果」と「期待してそう見える効果」を区別するために使われる道具だ。ここでは 246 の各タスクを、AI 使用可または使用不可へ無作為に割り当てた。そのため平均すれば、二群の系統的な違いは AI の有無だけになる。だから「AI を使う人が他の理由で速い／遅い」のを観察しただけではなく、AI が時間の変化を引き起こしたと言いやすくなる。

これは重要だ。AI コーディングによる生産性向上の一般的な証拠 —— 自己申告やベンチマーク得点 —— では、この因果関係を示せない。ベンチマークは実務ではなく、自己申告は、この研究が示すように、自信満々でも間違い得る。ここにいた開発者は、新しいおもちゃに戸惑う初心者ではない。大規模で成熟したオープンソースプロジェクトへ継続的に貢献し、コードをよく知り、ツールの使用経験もある人たちだった。

著者らが行ったこと

- **ランダム化比較試験**を行った（METR: Joel Becker、Nate Rush、Beth Barnes、David Rein）。熟練オープンソース開発者 16 人が、それぞれ日常的に貢献し、よく知る大規模リポジトリで作業した。特定プロジェクトへの経験は平均 5 年だった。

- **246 件の実務タスク** —— 各プロジェクトの実際の issue tracker から取ったバグ修正、機能追加、リファクタリング —— を使った。各タスクは無作為に「AI 可」または「AI 不可」へ割り当てられた。
- 「AI 可」で使えたのは **2025 年初頭のツール、Cursor Pro + Claude 3.5/3.7 Sonnet**。主要指標は各タスクの実際の完了時間だった。加えて、開発者の事前予測と事後推定、経済学・機械学習専門家の予測も集めた。

分かったこと

- **AI を使うとタスクは 19% 長くかかった**。速くではなく、遅くなった。95% 信頼区間はおよそ +2%〜+39% で、正確な大きさには幅があるが、方向はかなり堅い。
- **全員が逆を予測していた**。開発者は 24% の高速化、機械学習専門家は約 38%、経済学者は約 39% を予測した。三群とも大きな時間節約を期待したが、ストップウォッチは時間増加を示した。
- **認知ギャップ**。実際には遅くなった後でも、開発者は AI によって約 20% 速くなったと推定した。本人の感覚と時計の記録の間に、およそ 40 ポイントの差があった。
- **理由の候補は比較したが、証明はしていない**。著者らはいくつかの要因を挙げる。開発者が自分のコードベースを深く知っているため支援が追加できる余地が小さいこと、成熟プロジェクトには明文化されていないことも多い高い品質基準があること、リポジトリが大きくモデルの知らない文脈が多いこと、プロンプト作成に加え AI 出力のレビューや修正にも実時間がかかること。これらは手掛かりであって、確定原因ではない。

この研究からは言えないこと

- AI が**大多数の開発者を速くできない**とは示していない。著者ら自身が明記している。コードを知り尽くした 16 人の専門家は、平均的な開発者が平均的なタスクをする状況ではない。
- AI が役に立たない、あるいは**他の状況でも遅くする**とは示していない。コードベースに新しく入った人、ゼロから作る仕事、知らない言語、まったく別分野では結果が違い得る。
- ツールが**固定されたまま**とも仮定できない。使ったのは 2025 年初頭の Cursor と Claude 3.5/3.7 Sonnet。より良いツール、あるいは同じツールをより上手く使う方法でも、このまったく同じ状況の結果さえ変わり得ると著者らは明記する。
- これは**プレプリント**（2025 年 7 月公開、未査読）であり、著者らは実験アーティファクトを完全には排除できないとする。ただし、複数の解析で減速結果は維持された。
- 「時間以外で何かを得たのでは」と安心する読み方 —— 学習が増えた、気分が良かった、コード品質が上がった —— を裏付ける研究でもない。ここで本人が測ったと感じていた「高速化」こそ、実測データが否定した対象だ。

証拠をどう評価するか

- **設計は異例に誠実だ**。無作為化、実際のタスク、実際のリポジトリ、実時間の計測。AI コーディングについての主張が通常頼る自己申告やベンチマークリーダーボードから、大きく一歩進んでいる。19% の減速は著者らの頑健性チェックでも残った。
- **最も持ち運べる結果は認知ギャップ**。 AI による自分の高速化についての専門家の直感が、およそ 40 ポイント、楽観方向へ外れた。これは AI によるあらゆる自己申告の生産性向上を読む際の注意になる —— この研究自身の自己申告も含めて。
- **これは一時点のスナップショットで、傾向そのものではない**。 METR が 2026 年 2 月に行った、同種の開発者と新しいツールを使う追跡では、高速化の方向を示している。ごく大ざっぱに、再参加開発者で −18%、新規参加者で −4%。ただし著者らは証拠が弱いと注意する。参加者選択が歪み始めていたからだ。開発者は AI なしでの作業をますます断るようになり、報酬率が下がり、タスク選択も偏った。誠実な読み方は、景色が動いており、その動き自体も誇張せず報告されている、というものだ。

なぜ重要なのか

AI とプログラミングをめぐる議論の多くは、デモと感覚で進む。滑らかな画面録画、自信満々の主張、それに対する目を回す反応。珍しく、だから読む価値があるのは、誰かが退屈な実験をしたことだ。無作為化し、本物の仕事を計り、その後本人にどう感じたかを聞いた。答えは両陣営に居心地が悪い。AI が難しく慣れたコードでも専門家を一様にターボ化するという物語を破る。同時に「AI は開発者を遅くする。証明済み」という整った逆物語も破る。なぜなら同じチームの新しいデータがすでに逆方向を指すからだ。最も長く残る教訓は、小さく、とても人間的なものだ。作業者は、測定上遅くなりながら、速くなったと感じた。「速く感じる」は、速いことの証拠ではない。測ろう。

まとめ

METR のランダム化比較試験では、熟練オープンソース開発者 16 人が、自分たちのよく知るコードベースで 246 件の実務タスクを行い、その半分を無作為に 2025 年初頭の AI ツール（Cursor Pro + Claude 3.5/3.7 Sonnet）使用可とした。開発者は AI で作業時間が約 24% 短くなると予測したが、実際には完了時間が 19% 長くなった。しかも終了後も、自分たちは約 20% 速くなったと信じていた。この認知ギャップが、研究の最も鋭く頑健な核心である。ただし対象は 16 人、一つの狭い設定、2025 年初頭という固定スナップショットだ。著者らは、AI が大多数の開発者を助けないことを示す研究ではないと明記し、自分たちの 2026 年追跡データも、別の注意点を伴いながら高速化方向を示し始めている。真剣に受け取る価値のある慎重な測定であって、AI コーディングへの最終判決ではない。

出典

J. Becker, N. Rush, B. Barnes, D. Rein (METR), Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity, arXiv:2507.09089 [cs.AI] (2025)
https://arxiv.org/abs/2507.09089

METR, 'Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity' (study write-up, July 2025)
https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/

METR, developer-uplift follow-up (February 2026)
https://metr.org/blog/2026-02-24-uplift-update/