The Clean Paper

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

숙련 개발자들은 AI 로 더 빨라졌다고 느꼈지만 실제로는 측정상 더 느렸다—진짜 발견과, AI 코딩에 대한 판결이 아닌 이유

METR 의 무작위 대조시험에서 숙련 오픈소스 개발자 *16* 명이 자신들이 잘 아는 코드베이스의 실제 작업 *246* 개를 수행했고, 절반에는 *2025* 년 초 *AI* 도구 *(Cursor Pro* 와 *Claude 3.5/3.7 Sonnet)* 사용이 무작위로 허용됐다. 개발자들은 *AI* 가 작업 시간을 약 *24%* 줄일 것으로 예상했지만 실제로는 완료 시간이 *19%* 늘었다. 그런데도 이후에는 *AI* 가 약 *20%* 빠르게 했다고 믿었다. 이 체감과 측정의 간극이 연구의 선명하고 견고한 핵심이다. 하지만 개발자 *16* 명, 좁은 환경 하나, *2025* 년 초라는 고정된 시점이다. 저자들은 *AI* 가 대부분의 개발자에게 도움이 되지 않는다는 뜻이 아니라고 명시하며, 같은 팀의 *2026* 년 후속 결과는 자체적인 한계와 함께 이미 속도 향상 쪽을 가리킨다.

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

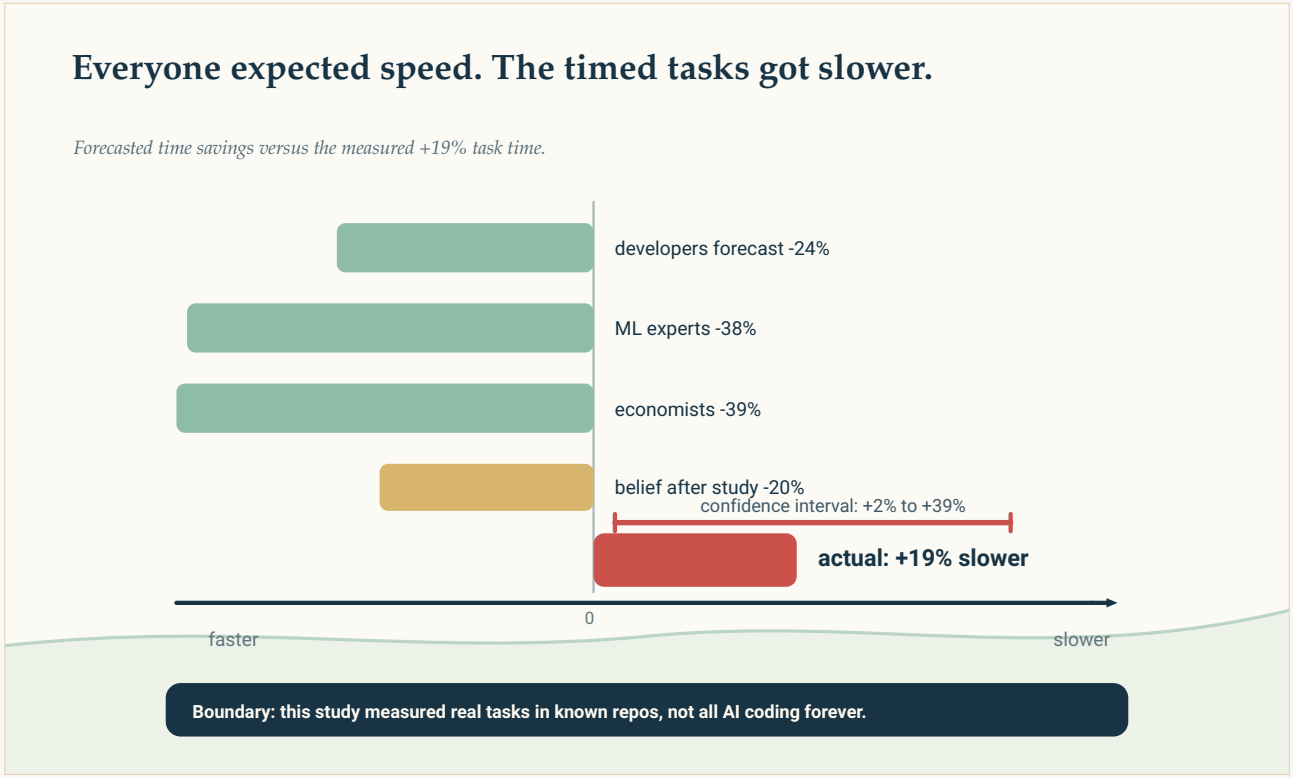
Paper: *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*, Joel Becker, Nate Rush, Beth Barnes, David Rein (METR), arXiv:2507.09089 [cs.AI] (preprint) · DOI: 10.48550/arXiv.2507.09089

숙련 개발자들은 AI 로 더 빨라졌다고 느꼈지만 실제로는 더 느렸다 — 진짜 발견과, AI 코딩 전체에 대한 판결이 아닌 이유

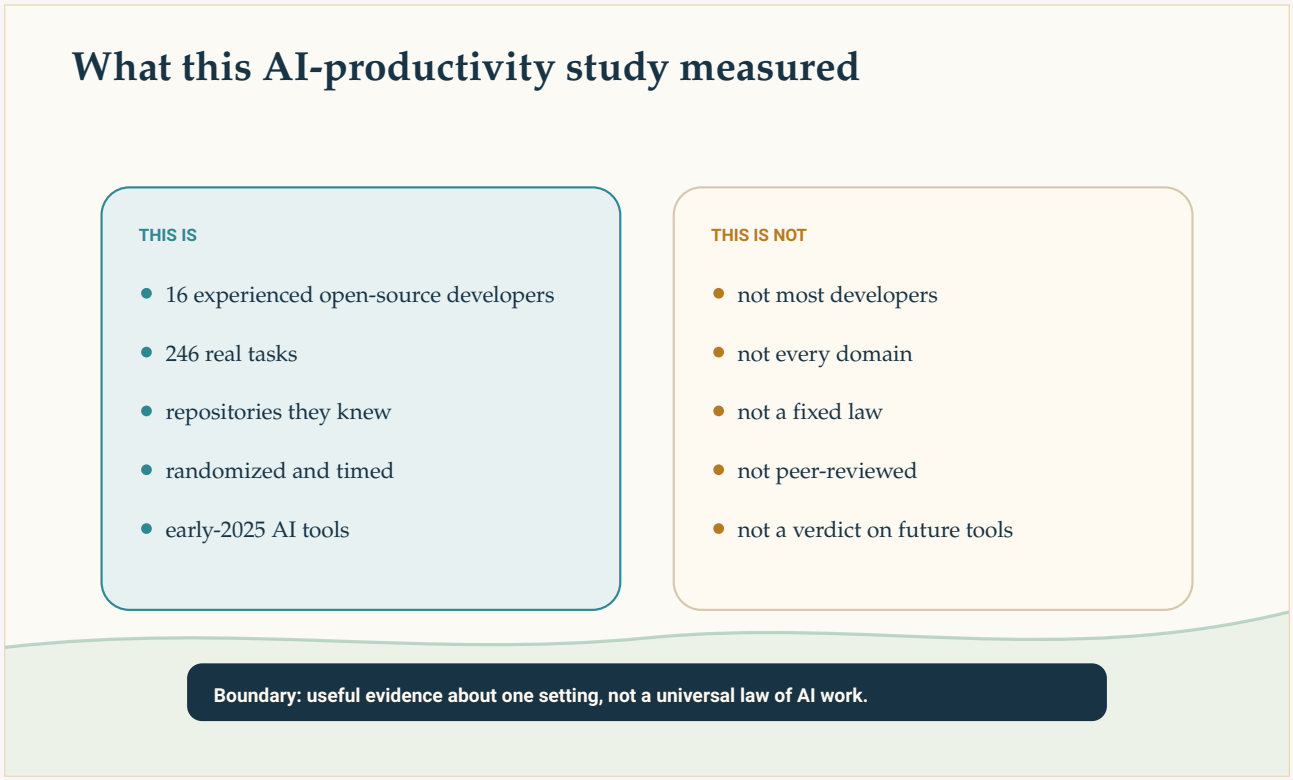
숙련 프로그래머에게 AI 코딩 도우미가 얼마나 시간을 줄여 주는지 물으면 흔히 숫자가 돌아온다. 20%, 30% 쯤 절약된다는 식이다. 경제학자나 머신러닝 연구자에게 물으면 그 숫자는 더 커진다. 2025 년 초 METR 연구팀은 느리고 비싼 방법을 택했다. 실제로 확인한 것이다. 대형 오픈소스 프로젝트를 잘 아는 숙련 개발자 16 명에게 그들이 익숙한 코드베이스의 실제 작업 246 개를 맡기고, 작업마다 무작위로 AI 도구 사용을 허용하거나 금지했다. 그리고 시간을 쟀다.

개발자들은 AI 가 작업 시간을 약 24% 줄일 것이라고 예상했다. 결과는 반대였다. AI 를 사용한 작업은 **19% 더 오래 걸렸다**. 더 흥미로운 부분은 그 다음이다. 작업을 모두 마친 뒤에도 같은 개발자들은 AI 가 자신들을 약 20% 더 빠르게 만들었다고 믿었다. 실제로는 느려졌는데 더 빨라졌다고 느꼈다. 이 두 숫자 사이의 거리가 이 연구에서 가장 흥미로운 결과다.

이는 실제 작업에서 꼼꼼하게 측정한 결과다. 하지만 동시에 개발자 16 명, 그들이 속속들이 아는 저장소, 2025 년 초의 도구라는 좁은 조건이다. “AI 는 개발자를 느리게 만든다” 라는 평면적인 결론은 아니다. 같은 연구팀의 이후 자료는 이미 반대 방향을 가리킨다.



모두가 AI 가 작업을 빠르게 할 것이라고 예상했다. 개발자 −24%, ML 전문가 −38%, 경제학자 −39%, 그리고 작업 후 개발자 스스로의 믿음도 −20% 였다. 실제 측정은 +19% 더 느렸고, 구간은 +2% 에서 +39% 였다. 느낀 속도와 측정된 속도의 차이가 핵심 결과다.



연구가 측정한 것: 숙련 오픈소스 개발자 16 명, 실제 작업 246 개, 익숙한 저장소, 2025 년 초 도구, 무작위 배정. 측정하지 않은 것: 대부분의 개발자, 다른 분야 전체, 변하지 않는 법칙, 동료평가를 마친 결론.

무엇을 측정했고, 무작위 시험이 왜 중요한가

무작위 대조시험 (randomized controlled trial, RCT) 은 의학에서 기대 효과와 실제 효과를 구분할 때 쓰는 도구다. 여기서는 246 개 작업 각각을 AI 사용 허용 또는 금지 조건에 무작위 배정했다. 따라서 평균적으로 두 집단 사이의 체계적인 차이는 AI 사용 여부뿐이 되도록 만든다. 그래서 단지 AI 를 찾는 사람이 원래 빠르거나 느린 것인지가 아니라, AI 가 작업 시간 변화의 원인이었다고 말할 수 있다. AI 코딩 생산성 주장에 흔히 쓰이는 자기보고나 벤치마크 점수는 이 일을 할 수 없다. 벤치마크는 실제 업무가 아니고, 이 연구가 보여 주듯 자기보고는 자신 있게 틀릴 수 있다. 여기 참여한 개발자들도 새 장난감을 서툴게 만지는 초보자가 아니었다. 자신들이 잘 아는 크고 성숙한 오픈소스 프로젝트의 기존 기여자들이었고, 도구 사용 경험도 일부 있었다.

연구진이 한 일

- **무작위 대조시험**을 수행했다 (METR: Joel Becker, Nate Rush, Beth Barnes, David Rein). 숙련 오픈소스 개발자 16 명이 각자 평소 기여하고 잘 아는 대형 저장소에서 작업했다. 해당 프로젝트에 참여한 기간은 평균 5 년이었다.
- 프로젝트의 실제 이슈 트래커에서 나온 버그 수정, 기능 추가, 리팩터링 등 **실제 작업 246 개**를 사용했다. 각 작업은 “AI 허용” 또는 “AI 금지” 에 무작위 배정됐다.
- “AI 허용” 은 **2025 년 초 도구인 Cursor Pro 와 Claude 3.5/3.7 Sonnet** 사용을 뜻했다. 주된 측정치는

작업당 실제 완료 시간이었다. 연구팀은 이와 함께 개발자의 사전 예측과 사후 추정, 경제학 및 머신러닝 전문가의 예측도 수집했다.

무엇을 발견했나

- **AI 를 쓰면 작업에 19% 더 오래 걸렸다.** 빨라진 것이 아니라 느려졌다. 95% 신뢰구간은 대략 +2% 에서 +39% 로, 정확한 크기에는 불확실성이 있어도 방향은 비교적 뚜렷하다.
- **모두가 반대를 예상했다.** 개발자는 24% 단축, 머신러닝 전문가는 약 38%, 경제학자는 약 39% 단축을 예상했다. 세 집단 모두 큰 시간 절약을 기대했지만 실제 시계는 시간이 더 들었다고 말했다.
- **인지의 간극이 있었다.** 실제로 더 느려진 뒤에도 개발자들은 AI 가 약 20% 빠르게 했다고 추정했다. 체감과 실제 측정 사이에 약 40% 포인트의 차이가 난 셈이다.
- **가능한 이유는 제시했지만 증명하지는 않았다.** 저자들은 몇 가지 후보를 검토한다. 이 개발자들은 자신의 코드베이스를 깊이 알고 있어 도우미가 추가할 가치가 적을 수 있다. 성숙한 프로젝트에는 높고 종종 암묵적인 품질 기준이 있다. 저장소는 크고 모델이 모르는 맥락이 많다. 프롬프트 작성, AI 출력 검토와 수정에도 실제 시간이 든다. 저자들은 이를 단서로 제시하지 확정된 설명으로 제시하지 않는다.

이것이 보여 주지 않는 것

- **AI 가 대부분의 개발자를 빠르게 만들지 못한다는 뜻이 아니다.** 저자들도 직접 말한다. 코드베이스를 거의 외우다시피 아는 전문가 16 명은 평균적인 작업을 하는 평균적인 개발자가 아니다.
- AI 가 쓸모없거나, 다른 환경에서도 사람을 느리게 만든다는 뜻이 **아니다**. 코드베이스에 새로 온 사람, 그린필드 작업, 익숙하지 않은 언어, 전혀 다른 분야에서는 결과가 다를 수 있다.
- 도구가 그대로 멈춰 있다는 뜻도 **아니다**. 연구는 2025 년 초 Cursor 와 Claude 3.5/3.7 Sonnet 을 다뤘다. 저자들은 더 좋은 도구나 더 나은 사용법이 정확히 같은 환경에서도 결과를 바꿀 수 있다고 명시한다.
- 이 연구는 **프리프린트**다 (2025 년 7 월 게시, 아직 동료평가 전). 저자들은 실험적 인공효과를 완전히 배제할 수 없다고 인정한다. 다만 여러 분석에서도 속도 저하 결과는 유지됐다.
- “대신 더 많이 배웠거나, 더 행복했거나, 더 좋은 코드를 썼을 것” 이라는 안심되는 해석을 허용하는 것도 **아니다**. 여기서 측정한 한 가지인 체감 속도 향상은 바로 데이터와 충돌한다.

근거는 얼마나 탄탄한가?

- **연구 설계는 보기 드물게 솔직하다.** 무작위 배정, 실제 작업, 실제 저장소, 실제 시간 측정이다. AI 코딩 주장 대부분이 기대는 자기보고와 벤치마크 순위표보다 큰 진전이다. 19% 속도 저하는 저자들의 강건성 점검에서도 유지됐다.
- **가장 널리 적용할 수 있는 결과는 인지의 간극이다.** 자신의 AI 생산성 향상에 관한 전문가의 직관이 낙관적인 방향으로 약 40% 포인트 틀렸다. 이는 AI 생산성에 관한 모든 자기보고를 볼 때 조심해야 한다는 경고다. 이 연구의 숫자 자체도 예외가 아니다.

- **추세가 아니라 한 시점의 스냅샷이다.** METR 이 2026 년 2 월 공개한 후속 결과는 더 최신 도구를 쓴 비슷한 개발자들에서 오히려 속도 향상을 가리킨다. 매우 대략적으로 재참여 개발자는 −18%, 신규 참여자는 −4% 였다. 하지만 저자들은 이를 약한 증거라고 경고한다. AI 없이 일하기를 거부하는 개발자가 늘었고, 보수가 낮아졌으며, 작업 선택도 치우쳤기 때문이다. 가장 정직한 해석은 상황이 움직이고 있고, 그 움직임 자체도 편향을 경계하며 보고되고 있다는 것이다.

왜 중요한가

AI 와 프로그래밍에 대한 논쟁은 대개 시연과 느낌으로 돌아간다. 매끈한 화면 녹화, 자신 있는 주장, 반대편의 냉소가 오간다. 이 연구가 읽을 가치가 있는 이유는 누군가 지루한 실험을 했기 때문이다. 무작위로 나누고, 실제 일을 재고, 끝난 뒤 당사자에게 어떻게 느꼈는지 물었다. 결과는 두 진영 모두에게 불편하다. AI 가 어렵고 익숙한 코드에서 전문가를 언제나 터보차저처럼 가속한다는 이야기를 찌른다. 동시에 "AI 가 개발자를 느리게 만든다는 것이 증명됐다" 는 단순한 반대 결론에도 제동을 건다. 같은 팀의 더 최신 자료가 이미 다른 방향을 향하고 있기 때문이다. 가장 오래 남을 교훈은 작고 인간적이다. 사람들은 측정상 더 느린데도 자신이 더 빨라졌다고 느꼈다. **"더 빨라진 느낌" 은 실제로 빨라졌다는 증거가 아니다. 재야 한다.**

핵심 요약

METR 은 무작위 대조시험에서 숙련 오픈소스 개발자 16 명에게 자신들이 잘 아는 코드베이스의 실제 작업 246 개를 수행하게 하고, 절반을 무작위로 2025 년 초 AI 도구 (Cursor Pro 와 Claude 3.5/3.7 Sonnet) 사용 허용 조건에 배정했다. 개발자들은 AI 가 작업 시간을 약 24% 줄일 것이라고 예상했지만 실제로는 완료 시간이 19% 늘었다. 그런데도 이후에는 여전히 AI 가 약 20% 빠르게 했다고 믿었다. 이 인지의 간극이 연구의 가장 선명하고 견고한 핵심이다. 하지만 개발자 16 명, 좁은 한 환경, 2025 년 초라는 고정된 시점이다. 저자들은 이 결과가 AI 가 대부분의 개발자에게 도움이 되지 않는다는 뜻은 아니라고 명시하며, 같은 팀의 2026 년 후속 자료는 여러 단서와 함께 이미 속도 향상 쪽을 가리킨다. **진지하게 받아들일 가치가 있는 정밀한 측정이지, AI 코딩 전체에 대한 판결은 아니다.**

출처

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/