

Vibe Coding, CS Knowledge, and DHH's Claim

Julien Laurenceau | Published 2026-09-06 | Updated 2026-10-21

<https://pepitedata.com/dhh-followup-vibe-coding-data/>

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



Does knowing computer science still help when an AI writes the code? A CHI 2026 study gives a useful, limited answer. Higher CS achievement and stronger writing were associated with better results on a set of vibe-coding tasks. That deserves attention without turning a student study into a verdict on every engineer or every AI workflow.

In my [earlier article on DHH's AI-coding argument](#), I separated programming knowledge from the habit of prescribing an implementation. This follow-up examines what the evidence adds, where my earlier interpretation went too far, and how I would evaluate the question inside an engineering team.

Revised October 21, 2026. The paper was submitted to arXiv on March 14, 2026. This update corrects the study-design description and the statistical comparison, removes unverified social-media framing, and separates reported results from my recommendations.

What did the study measure?

The study by Sverrir Thorgeirsson, Theo B. Weidmann and Zhendong Su was preregistered and cross-sectional, with 100 university students. Participants completed CS, reasoning and writing assessments, plus three timed GUI-building tasks. They could prompt, run and inspect applications, but could not view or edit source code. One author graded application features blind to the other participant measures.

A controlled environment is not the same thing as a randomized intervention. The researchers measured existing differences between participants; they did not randomly assign people to acquire CS knowledge or writing skill. A relationship between scores therefore does not establish what would happen if a team introduced a particular training course.

Both skills were associated with performance

CS achievement and writing both showed positive associations with vibe-coding performance. The raw CS correlation was larger, while the reasoning-adjusted comparison gave a different view of each relationship. These are estimates within this sample and assessment. Neither a correlation coefficient nor a significance threshold is a universal ranking of which skills an engineering team should value.

Assessment	Raw Pearson r	Partial r, controlling for reasoning
CS achievement	0.386, p < 0.001	0.281, p = 0.005
Writing	0.290, p = 0.003	0.186, p = 0.066

Source: Table 3 of the paper, N = 100. Partial correlations control for the ICAR16 cognitive assessment; they are not the two-predictor regression below.

The p = 0.066 result does not establish that writing has no value. It says that this adjusted association did not meet the chosen 0.05 significance threshold. A significant result for one predictor and a nonsignificant result for another also does not, by itself, test whether their effects differ.



Table 3, Thorgeirsson, Weidmann and Su, CHI 2026. Both raw associations are positive. These coefficients are not success rates, causal effects or evidence that one skill makes a person twice as effective.

What does roughly twice actually mean?

The paper's roughly two-to-one comparison concerns incremental explained variance in a regression using CS and writing scores. It does not compare success rates or the two raw correlations. Keeping the statistic and its model attached to the claim prevents a narrow result from becoming an unsupported promise about developer productivity.

Model comparison	Reported result	Meaning
Add CS after writing	Delta $R^2 = 0.125$	12.5 percentage points of additional explained variance
Add writing after CS	Delta $R^2 = 0.059$	5.9 percentage points of additional explained variance
Both predictors together	$R^2 = 0.208$	20.8% of outcome variance explained within the sample

Source: Section 8.2. The two-predictor model uses CS and writing, without the reasoning covariate. The reported increments are rounded.

The arithmetic behind the wording is $0.125 / 0.059$, approximately 2.12. Dividing the raw correlations instead gives $0.386 / 0.290$, approximately 1.33, and even that ratio is not a productivity multiplier. The earlier image caption on this page conflated the comparisons; it was wrong.

Explained variance describes how the fitted model accounts for differences in observed scores. It does not mean 20.8% of a successful app was written by those skills, or that changing either skill would deliver a predictable improvement. It also does not establish predictive accuracy on a new population without validation.

Does this settle DHH's claim?

This study does not directly test whether programmers underperform non-programmers on product work, or whether prescriptive prompts hurt results. Those questions need comparisons designed for those claims. Positive associations with a CS assessment challenge a blanket dismissal of technical knowledge; they neither validate every experienced engineer's habits nor settle how much implementation detail an agent needs.

In the [Lex Fridman interview](#), around 52 to 55 minutes, DHH describes his own shift from prescribing implementation paths to describing desired outcomes. Asked whether programmers can do worse than non-programmers on some problems, he points to product-management skills that programmers do not all share equally. This is practitioner testimony about a workflow, not a measured population effect.

The distinction I find useful is between knowledge and how that knowledge is applied. A constraint can encode an essential requirement, such as preserving a public API. It can also encode a preference that needlessly restricts the solution. Calling both cases over-specification hides the decision the team needs to make.

I would no longer describe DHH's narrow claim as something this study shows to be true. It remains a hypothesis worth testing. Equally, an association between knowledge and performance is not evidence that expertise can never become counterproductive in a particular situation.

Which conclusions go beyond the evidence?

The strongest claims in this debate need evidence beyond a cross-sectional task assessment. Hiring performance, long-term skill retention and safe operation of production software are different outcomes. My practical concern is the change in question: an app score can inform a discussion about those outcomes, but it cannot stand in for measuring them.

Claim	Missing comparison	Evidence to seek
AI use erodes engineering skill	Change in skill over time and a suitable comparison condition	Repeated assessments with an explicit identification strategy

Claim	Missing comparison	Evidence to seek
Less detailed prompts outperform detailed ones	Prompt strategies applied to comparable work	Repeated trials with controlled conditions and outcome criteria
A hiring screen predicts production performance	Performance on the job, including maintenance and diagnosis	Validation against role-relevant work samples and subsequent outcomes

My evaluation checklist, not additional findings reported by the study.

The paper also reports exploratory negative associations between self-reported LLM use and writing and task scores. That cannot identify a causal direction. People who struggle may use an assistant more; using it may change how they practice; other differences may affect both. Without a design that separates those explanations, an erosion claim is premature.

I also removed the earlier blanket assertion about how senior engineers work. A job title does not guarantee review discipline. For production changes, I want evidence of testing, code review, failure handling and ownership. That is my engineering recommendation, not a finding established by these student tasks.



A result can be useful without supporting a verdict on every developer. Match each claim to the comparison that would test it.

How I would test the workflow

I would compare prompting strategies on matched tasks with the same model, tools and acceptance criteria. One easy task with a short prompt and one difficult task with a long specification cannot establish which strategy works better. The goal is an operational decision for the team's workload, with uncertainty reported, rather than a universal winner.

1. Choose several representative tasks and define the acceptance tests before running either strategy. Use synthetic or approved data; keep customer material out of unauthorized model services.
2. Prepare an outcome-focused prompt and an implementation-focused prompt for each task. Keep required behavior, security constraints and available context equivalent.

3. Run fresh sessions with the same model version and tools. Repeat trials and randomize or counterbalance assignment and order to reduce learning and task-selection effects.
4. Evaluate outputs without revealing the prompt condition where feasible. Record correctness, review and rework time, usage cost, and maintenance effort for a later change.
5. Inspect failures and uncertainty before changing the process. A local comparison informs that setting; it does not establish how every engineer or future model will behave.

For hiring, I would ask a candidate to explain a generated design, identify a failure case and improve the tests. I would assess clear communication alongside technical understanding. Treat that as a proposed work sample to validate for the role, not a hiring rule proved by this paper.

FAQ

The useful distinction is between an association, a causal effect and a recommendation. Research can motivate a workflow change without proving that change will help every team. For decisions about agents, I would state the target outcome first, then ask whether the available evidence actually measures it under comparable conditions.

Did the study find that writing does not matter?

No. Writing had a positive raw association and contributed additional explained variance beyond CS achievement. The reasoning-adjusted result asks a different statistical question. Failing a significance threshold does not prove that a skill has no value.

Does roughly twice mean twice the productivity?

No. The comparison concerns increments in explained variance in one regression model. It is not a ratio of success rates, time saved or developer productivity. Any quotation of the comparison should identify the statistic and model.

Does this prove experienced programmers always do better?

No. An association between an assessment score and task performance does not establish a universal rule about individuals or professional experience. It cannot rule out context-specific disadvantages or show that one prompting habit is better than another.

Should teams stop reviewing AI-generated code?

No. I recommend review and testing appropriate to the consequences of a change. A working demonstration does not establish security, maintainability or correct failure behavior. Responsibility for accepting a production change stays with the team that operates it.

Sources and related reading

The statistical results come from the CHI 2026 paper; the workflow account comes from DHH's interview. My interpretation and proposed team evaluation are identified separately. The original research and transcript are the sources to check when reusing a claim, rather than a social-media summary or an image detached from its caption.

- Thorgeirsson, Weidmann and Su, [Computer Science Achievement and Writing Skills Predict Vibe Coding Proficiency](#), CHI 2026. [Author manuscript and submission history](#); results in Table 3 and Section 8.2, limitations in Section 10.2.
- [DHH on Lex Fridman Podcast #501](#), especially the discussion around 52 to 55 minutes.

- [My earlier reading of DHH's claim about programmers and AI coding](#)
- [Why a product trend is not a benchmark](#)
- [AI vulnerability scanning and the limits of a headline result](#)

I have no affiliation with the study's authors or institutions. No endorsement is implied. This article uses paraphrases and original analysis rather than extended quotations.

If you need an independent review of a data or AI platform, the [data platform audit](#) covers implementation, architecture, performance and operational behavior. [Book a free 15-minute intro call](#) to discuss the question you need answered.

PDF snapshot of the public article. The online version is authoritative; links remain clickable.