

# Beyond the Hype: Studying the Impact of AI Assistants on Software Development

Prof. Dr. Sebastian Baltes



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

**SE@UHD**  
Software Engineering

# “Hi, my name is Sebastian and I’m a pracademic”

<https://en.wikipedia.org/wiki/Pracademic>

Industry



THE UNIVERSITY  
of ADELAIDE



UNIVERSITÄT  
BAYREUTH



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

Academia

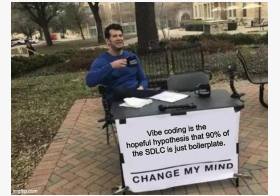
# Four Takeaways of this Talk



One does not simply **measure** software development **efficiency/productivity**.



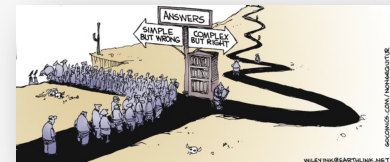
“**Vibe coding**” without following software engineering best practices is **bound to fail**.



“**Trusting AI**” entails much more than just tool **reliance**.



**GenAI** makes good empirical **research harder** rather than easier.



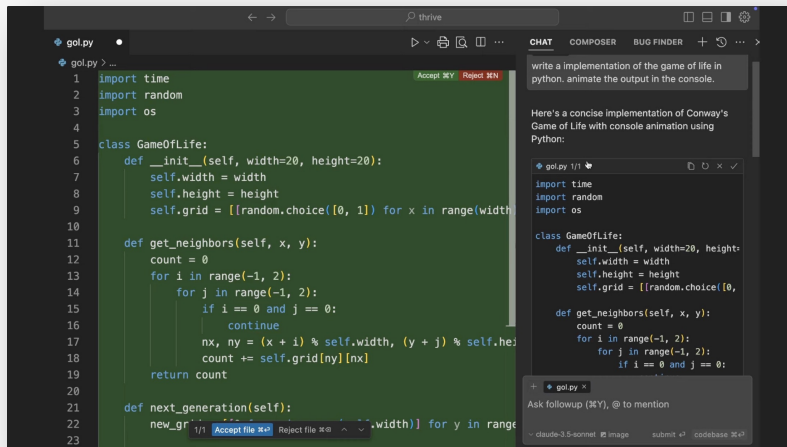
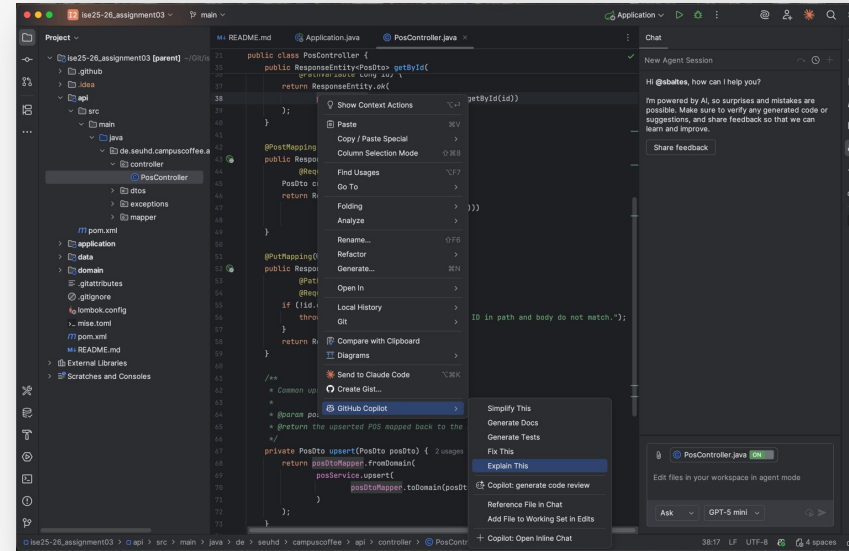
# Assessing the impact of **AI Assistants** on **Development Productivity**



# AI Assistants?

# Editor-integrated and chat-based AI assistants for software development.

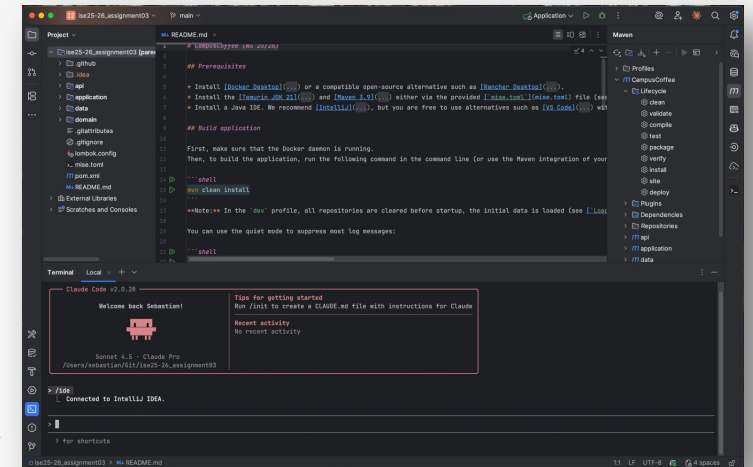
 **GitHub Copilot**



GenAI-  
centric **IDEs**



"Agentic" AI  
assistants



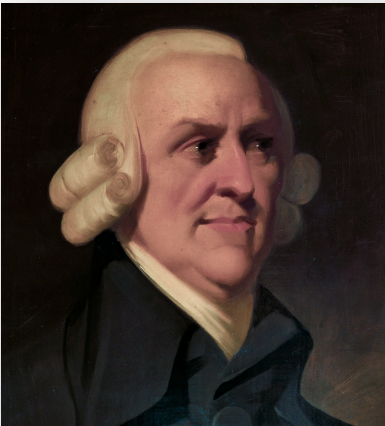
# Development Productivity?

# Productivity?

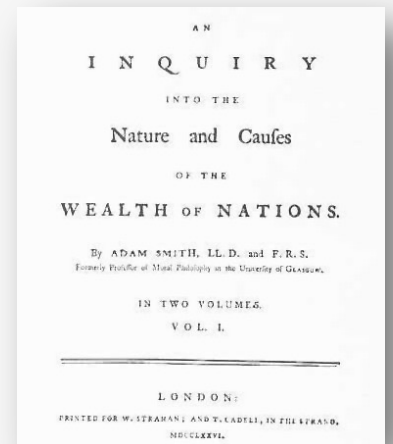
# Origins of the Term “Productivity”

*“The greatest improvements in the **productive powers of labour** [...] seem to have been the effects of the **division of labour**.”*

Adam Smith: *The Wealth of Nations* (1776)



[https://www.gutenberg.org/files/38194/38194-h/38194-h.htm#Page\\_2](https://www.gutenberg.org/files/38194/38194-h/38194-h.htm#Page_2)



# Knowledge Work: “Historic” Productivity Gains

## Toward Characterizing the Productivity Benefits of Very Large Displays

**Mary Czerwinski, Greg Smith, Tim Regan, Brian Meyers, George Robertson and Gary Starkweather**

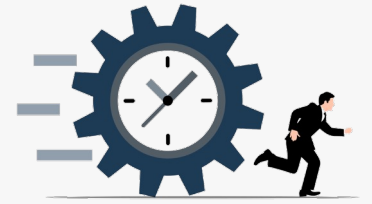
Microsoft Research, One Microsoft Way, Redmond, WA, 98052, USA  
marycz@microsoft.com

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/interact2003-productivitylargedisplays.pdf>

**2003: ~10% “productivity” increase** by using 42-inch instead of 15-inch **display**.  
(set of tasks, comparison of completion time)

# Development Productivity?

# Efficiency vs. Productivity



$$\text{Productivity} = \frac{\text{Output}}{\text{Input}}$$

$$\text{Productivity} = \frac{\text{\#New Features Deployed}}{\text{\#Person Hours}}$$

- **Input:** Time, Money, Resources.      **Output:** Value added.
- **Value:** Can be *functional* (new features) or *non-functional* (improved maintainability, usability, performance, etc.).
- **Productivity** ~ maximizing output.
- **Efficiency** ~ minimizing input.
- **Developer Experience:** Broader concept, less focus in input and output, includes aspects such as *well-being* and *satisfaction*.





# Yes, you can measure software developer productivity

August 17, 2023 | Article

<https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/yes-you-can-measure-software-developer-productivity#/>

# Measuring developer productivity? A response to McKinsey

The consultancy giant has devised a methodology they claim can measure software developer productivity. But that measurement comes at a high price – and we offer a more sensible approach. Part 1.



GERGELY OROSZ AND KENT BECK

AUG 29, 2023

<https://newsletter.pragmaticengineer.com/p/measuring-developer-productivity>

<https://newsletter.pragmaticengineer.com/p/measuring-developer-productivity-part-2>

To cut to the chase, I see two main planks to your thesis, which I will return to as we go through the content, and which are both erroneous:

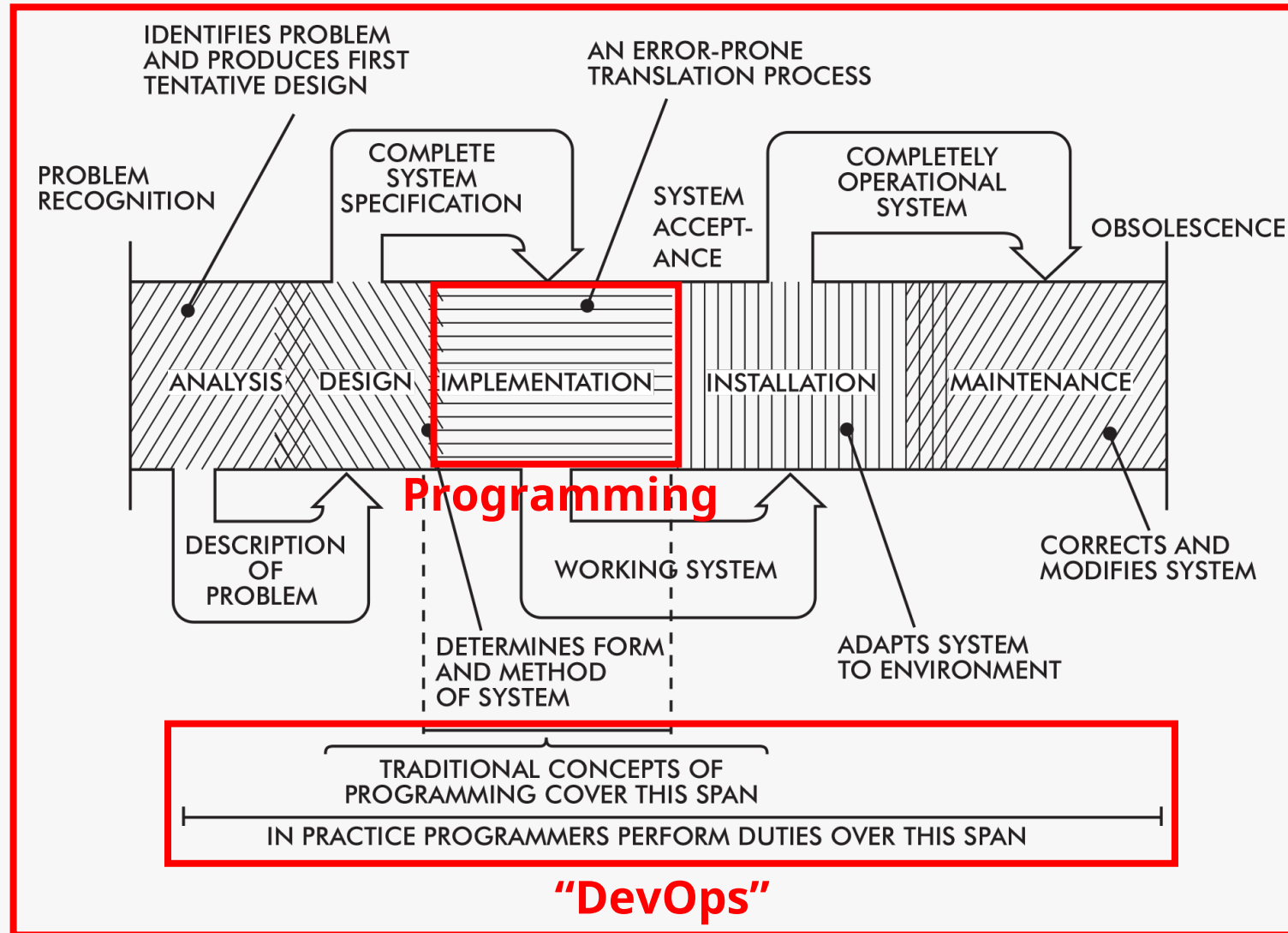
1. **Software development is a reducible activity**, and can be measured with reductionist tools.
2. **Software development is primarily about coding**, and anything other than typing code into a computer terminal is waste which we should seek to eliminate.

I hope to explain why both of these are incorrect as we go through.

**Daniel Terhorst-North**

<https://dannorth.net/mckinsey-review/>

# SE Has Never Been Just About Coding!



## Software Engineering

# “State of the Art” in Software Engineering as of 2019

## No Single Metric Captures Productivity

Ciera Jaspan, Google, USA

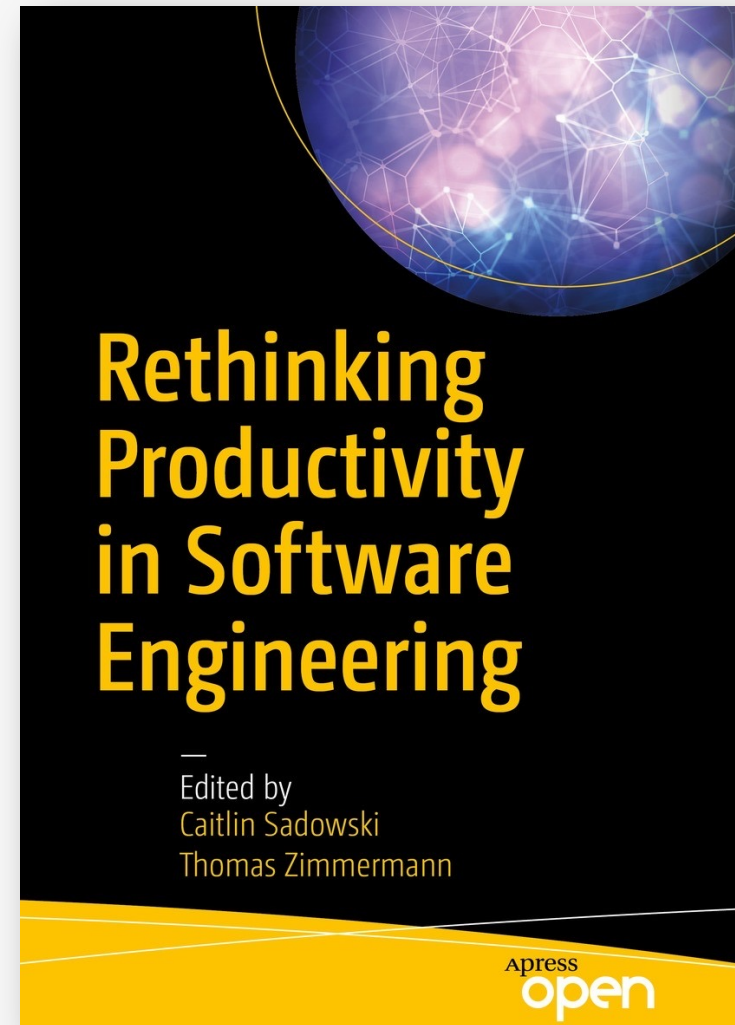
Caitlin Sadowski, Google, USA

*“Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs.”*

—Bill Gates

## Why We Should Not Measure Productivity

Amy J. Ko, University of Washington, USA



<https://link.springer.com/book/10.1007/978-1-4842-4221-6>



**ONE DOES NOT SIMPLY**

**MEASURE DEVELOPMENT PRODUCTIVITY**

# Current Discourse on AI for SE



# Perception and Usage Data: GitHub Copilot

DOI:10.1145/3633453

**Case study asks Copilot users about its impact on their productivity, and seeks to find their perceptions mirrored in user data.**

BY ALBERT ZIEGLER, EIRINI KALLIAMVAKOU, X. ALICE LI, ANDREW RICE, DEVON RIFKIN, SHAWN SIMISTER, GANESH SITTAMPALAM, AND EDWARD AFTANDILIAN

## Measuring GitHub Copilot's Impact on Productivity

<https://dl.acm.org/doi/10.1145/3633453>

- **Publication:**  
March 2024 (peer-reviewed)
- **Data Collection:**  
Feb-Mar 2022
- **Participants:**  
2,047 Copilot users
- **Goal:**  
Correlating usage data and survey responses

**Experienced developers who are already highly skilled are less likely to write better code with Copilot, but Copilot can assist their productivity in other ways.**



# Perception and Usage Data: GitHub Copilot

**Table 1. Developer usage events collected by GitHub Copilot.**

Opportunity	A heuristic-based determination by the IDE and the plug-in that a completion might be appropriate at this point in the code (for example, the cursor is not in the middle of a word)
Shown	Completion shown to the developer
Accepted	Completion accepted by the developer for inclusion in the source file
Accepted char	The number of characters in an accepted completion
Mostly unchanged X	Completion persisting in source code with limited modifications (Levenshtein distance less than 33%) after X seconds, where we consider a duration of 30, 120, 300, and 600 seconds
Unchanged X	Completion persisting in source code unmodified after X seconds.
(Active) hour	An hour during which the developer was using their IDE with the plug-in active

**Table 2. The core set of measurements considered in this article.**

Natural name	Explanation
Shown rate	Ratio of completion opportunities that resulted in a completion being shown to the user
Acceptance rate	Ratio of shown completions accepted by the user
Persistence rate	Ratio of accepted completions unchanged after 30, 120, 300, and 600 seconds
Fuzzy persistence rate	Ratio of accepted completions mostly unchanged after 30, 120, 300, and 600 seconds
Efficiency	Ratio of completion opportunities that resulted in a completion accepted and unchanged after 30, 120, 300, and 600 seconds
Contribution speed	Number of characters in accepted completions per distinct, active hour
Acceptance frequency	Number of accepted completions per distinct, active hour
Persistence frequency	Number of unchanged completions per distinct, active hour
Total volume	Total number of completions shown to the user
Loquaciousness	Number of shown completions per distinct, active hour
Eagerness	Number of shown completions per opportunity

## C Aspects of Productivity Measured In The Survey

This table shows the relationship between the survey statements, the metrics and the different dimension of the SPACE framework [1].

Survey statements	Productivity aspect	Code	Metric name
"I am more productive when using GitHub Copilot"	Perceived productivity		more_productive
"I feel more fulfilled with my job when using GitHub Copilot."			more_fulfilled
"I find myself less frustrated during coding sessions when using GitHub Copilot."	Satisfaction and well-being	S	less_frustrated
"I can focus on more satisfying work when using GitHub Copilot."			focus_satisfying
"While working with an unfamiliar language, I make progress faster when using GitHub Copilot."	Performance	P	unfamiliar_progress
"The code I write using GitHub Copilot is better than the code I would have written without GitHub Copilot."			better_code
n/a	Activity	A	n/a
"I learn from the suggestions GitHub Copilot shows me."	Communication and col-laboration [2]	C	learn_from
"Using GitHub Copilot helps me stay in the flow."			stay_in_flow
"I complete tasks faster when using GitHub Copilot."			tasks_faster
"I complete repetitive programming tasks faster when using GitHub Copilot."			repetitive_faster
"I spend less mental effort on repetitive programming tasks when using GitHub Copilot."	Efficiency and flow	E	less_effort_repetitive
"I spend less time searching for information or examples when using GitHub Copilot."			less_time_searching



# Self-reports: Correlation Study Across 3 Companies



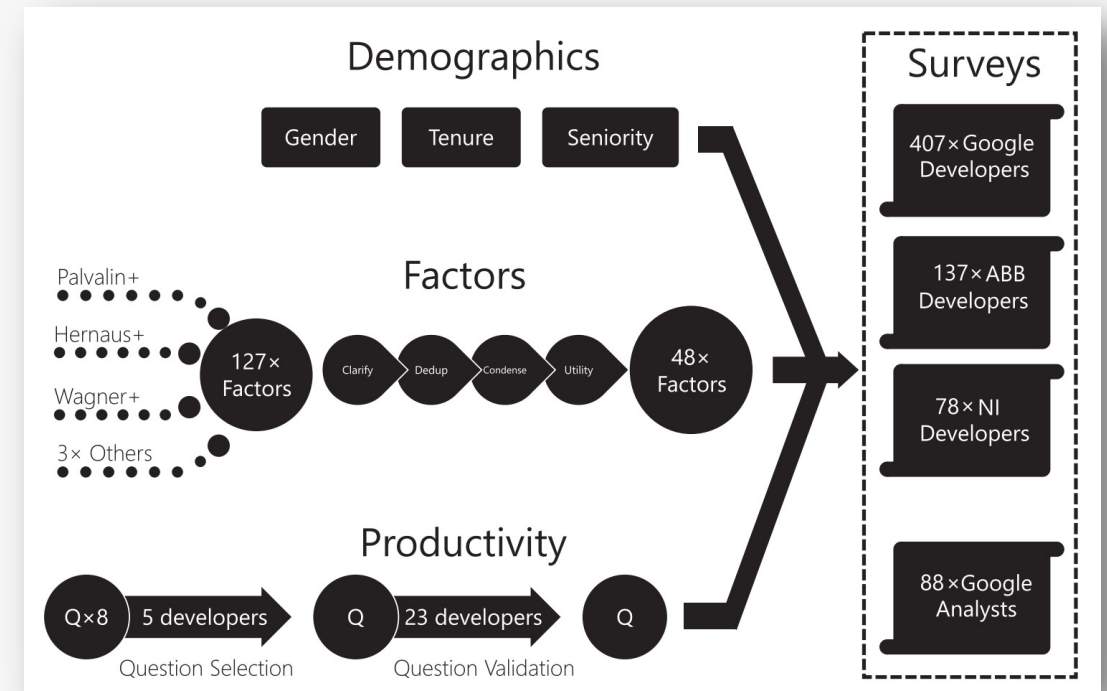
## What Predicts Software Developers' Productivity?

Emerson Murphy-Hill <sup>id</sup>, Ciera Jaspan <sup>id</sup>, Caitlin Sadowski, David Shepherd <sup>id</sup>, Michael Phillips <sup>id</sup>,  
Collin Winter, Andrea Knight, Edward Smith, and Matthew Jorde

<https://ieeexplore.ieee.org/document/8643844>

“Our results suggest that the **factors that most strongly correlate with self-rated productivity were non-technical factors**, such as job enthusiasm, peer support for new ideas, and receiving useful feedback about job performance. [...] our results also suggest that software developers' self-rated productivity is more **strongly related to task variety and ability to work remotely.**”

- **Publication:**  
February 2019 (peer-reviewed)
- **Participants:**  
622 software developers at Microsoft, ABB, National Instruments



# Controlled Experiment: Impact on Dev Activity

## The Effects of Generative AI on High Skilled Work: Evidence from Three Field Experiments with Software Developers

22 Pages • Posted: 5 Sep 2024

Zheyuan (Kevin) Cui

Princeton University - Bendheim Center for Finance

Mert Demirer

Massachusetts Institute of Technology (MIT)

Sonia Jaffe

Microsoft Research

Leon Musolff

University of Pennsylvania - Business & Public Policy Department

Sida Peng

Microsoft Corporation

Tobias Salz

Massachusetts Institute of Technology (MIT); National Bureau of Economic Research (NBER)

Date Written: September 03, 2024

[https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=4945566](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4945566)

- **Publication:**  
September 2024 (preprint)
- **Data Collection:**  
Sep 22-May 23 (Microsoft)   
Jul 23-Dec 23 (Accenture)   
Sep 23 (Anonymous Company)
- **Participants:**  
1,746 developers (Microsoft)  
320 (Accenture)  
3,054 (Anonymous Company)
- **Study Setup:**  
Microsoft/Accenture:    
Control vs. treatment groups  
A. Company:  
Focus on staggered roll-out

# Controlled Experiment: Impact on Dev Activity



## Metrics:

- #Pull Requests
- #Commits
- #Builds
- Build Success Rate

Outcome	Microsoft	Accenture	Anon. Comp.	Pooled
Pull Requests	27.38** (12.88)	17.94 (18.72)	54.03 (42.63)	26.08** (10.3)
Commits	18.32 (11.25)	-4.48 (21.88)	- -	13.55 (10.0)
Builds	23.19 (14.20)	92.40*** (26.78)	- -	38.38*** (12.55)
Build Success Rate	-1.34 (4.23)	-17.40** (7.12)	- -	-5.53 (3.64)
N Developers	1,521	316	3,030	4,867
N Clusters	690	316	432	1,438

Table 2: Experiment-by-Experiment Results (Weighted IV).

*Notes:* This table provides estimates of the effect of GitHub Copilot adoption on the number of Pull Requests, Commits and Successful Builds across three experiments at Microsoft, Accenture, and Anonymous Company. Each entry corresponds to an estimate of  $\beta$  in (1) expressed as a percentage of the control mean. Standard errors are clustered at the level of treatment assignment, which varies across experiments (Microsoft: mixed team-level and individual assignment; Accenture: individual assignment; Anonymous Company: team-level assignment.)

\*\* difference statistically significant

# Discussion



## 6 Conclusion

To summarize, we find that usage of a generative AI code suggestion tool increases software developer productivity by 26.08% (SE: 10.3%). This estimate is based on observing, partly over years, the output of almost five thousand software developers at three different companies as part of their regular job, which strongly supports its external validity.

## Really?

### Metrics:

- **#Pull Requests** → not controlled for size, complexity, etc.
- **#Commits** → not controlled for size, complexity, etc.
- **#Builds** → highest difference of all metrics, what does this tell us?
- **Build Success Rate** → decreased

Outcome	Microsoft	Accenture	Anon. Comp.	Pooled
Pull Requests	27.38** (12.88)	17.94 (18.72)	54.03 (42.63)	26.08** (10.3)

# Is “More Code” Always Better?



**Adam Tornhill** 

@AdamTornhill



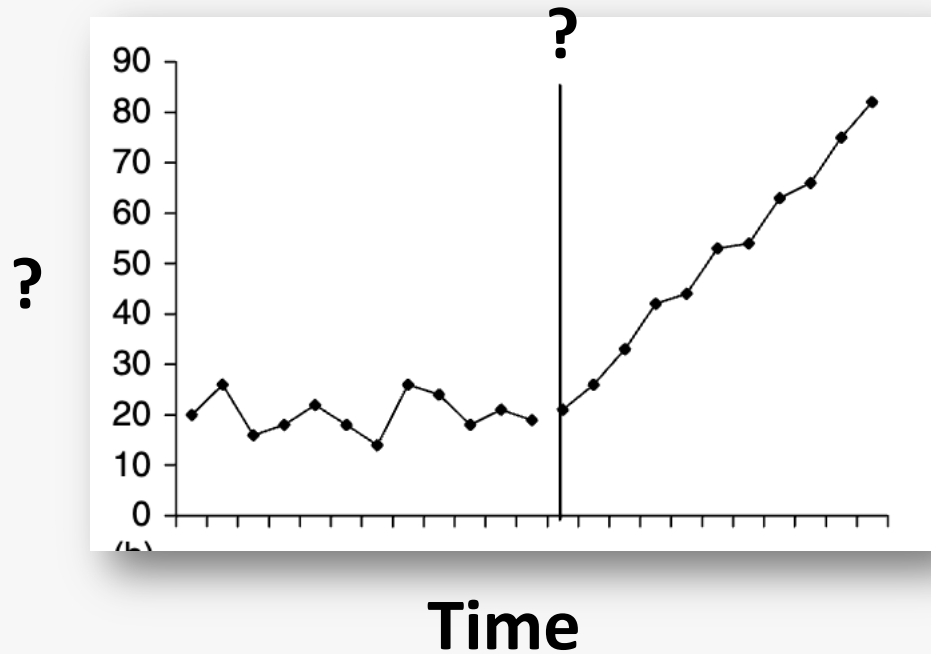
The main challenge with AI assisted programming is that it becomes so easy to generate a lot of code which shouldn't have been written in the first place.

9:05 PM · Nov 28, 2023 · **11.5K** Views

<https://x.com/AdamTornhill/status/1729592297887502611>

# How can we study GenAI adoption in open-source projects?

# Quasi-Experimental Time Series Analysis



## Interrupted Time Series Design

The interrupted time-series design [2] provides a method for researchers to examine the effect of an intervention on a single case, where the case may be a group or an individual. The basic interrupted time-

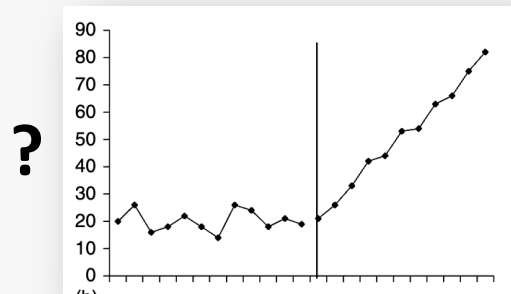
<https://doi.org/10.1002/0470013192.bsa312>

Selecting and Improving Quasi-Experimental Designs in Effectiveness and Implementation Research

[Margaret A. Handley](#),<sup>1,2</sup> [Courtney Lyles](#),<sup>2</sup> [Charles McCulloch](#),<sup>1</sup> and [Adithya Cattamanchi](#)<sup>3</sup>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8011057/>

# What to measure over time?

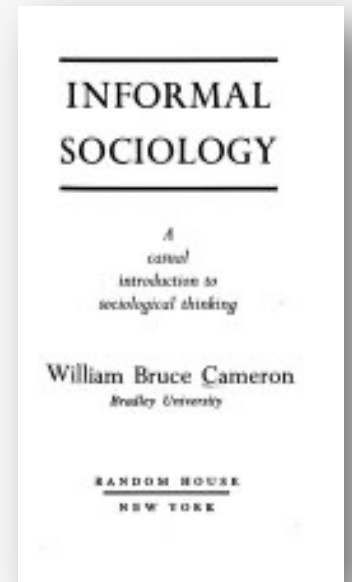




# General Limitation of Quantitative Measures

*"...not everything that can be counted counts,  
and **not everything that counts can be  
counted.**"*

William Bruce Cameron: Informal Sociology, a casual introduction to sociological thinking, p. 13, (1963).



# GitClear Report: Code Churn



## Coding on Copilot

*2023 Data Shows Downward Pressure on  
Code Quality*

*150m lines of analyzed code + projections for 2024*

## Burgeoning Churn

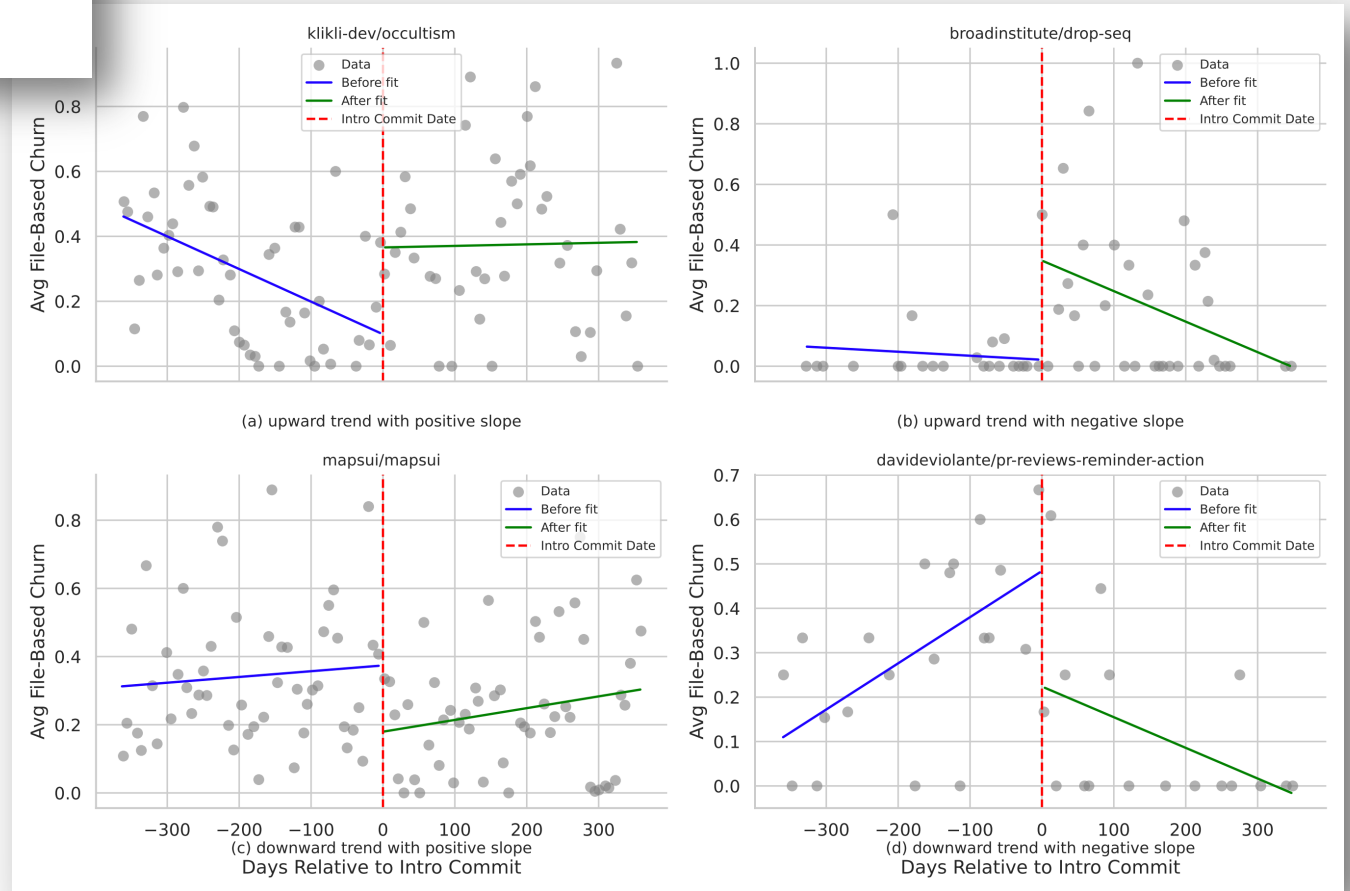
Recall that "Churn" is the percentage of code that was pushed to the repo, then subsequently reverted, removed or updated within 2 weeks. This was a relatively infrequent outcome when developers authored all their own code -- only 3-4% of code was churned prior to 2023, although there is also a hint of the coming uptick in 2022, when Churn jumped 9%. 2022 was the first year Copilot was available in beta, and the year that ChatGPT became available.

<https://gitclear-public.s3.us-west-2.amazonaws.com/Coding-on-Copilot-2024-Developer-Research.pdf>

# Self-Admitted GenAI Usage in Open-Source Software

Tao Xiao, Youmei Fan, Fabio Calefato, Christoph Treude, Raula Gaikovina Kula, Hideaki Hata, Sebastian Baltes ✉

We could **not** find a **general trend** of **higher code-churn** in GitHub projects with self-admitted GenAI usage.



<https://arxiv.org/pdf/2507.10422>

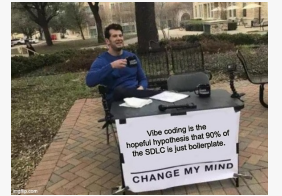
# Four Takeaways of this Talk



One does not simply **measure** software development **efficiency/productivity**.



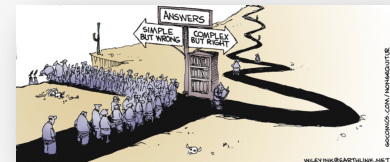
“**Vibe coding**” without following software engineering best practices is **bound to fail**.



“**Trusting AI**” entails much more than just tool **reliance**.



**GenAI** makes good empirical **research harder** rather than easier.




# Vibe Coding: The Tweet That Coined The Term



<https://x.com/karpathy/status/1886192184808149383>

# Vibe Coding: The Result

 **r/cursor** · 3 mo. ago  
Forsaken\_Space\_2120



**Cursor f\*ck up my 4 months of works**


**Question**


Disclaimer, I'm a moron who worked on the same project without thinking about the risk that Cursor could break everything. Yesterday, Cursor (even though I only asked it to feed a view on my UI) destroyed months of development.


My question: How do you back up your projects/versions to ensure that the next action on cursor is reversible? Ops!


Also, I know that while I'm the concern, cursor isn't the only culprit, it's also Claude (while good overall) still has some flaws

 182 

 257




 Share




 Report




Join the conversation


Sort by: Best ▾

 **whatisthereason** · 3mo ago



No version control use for 4 months?




  44 


 Reply  Award  Share ...

 **djimboboom** · 3mo ago

Does it terrify anyone else that there is an entire cohort of new engineers who are getting into programming because of AI, but missing these absolute basic bare necessities?

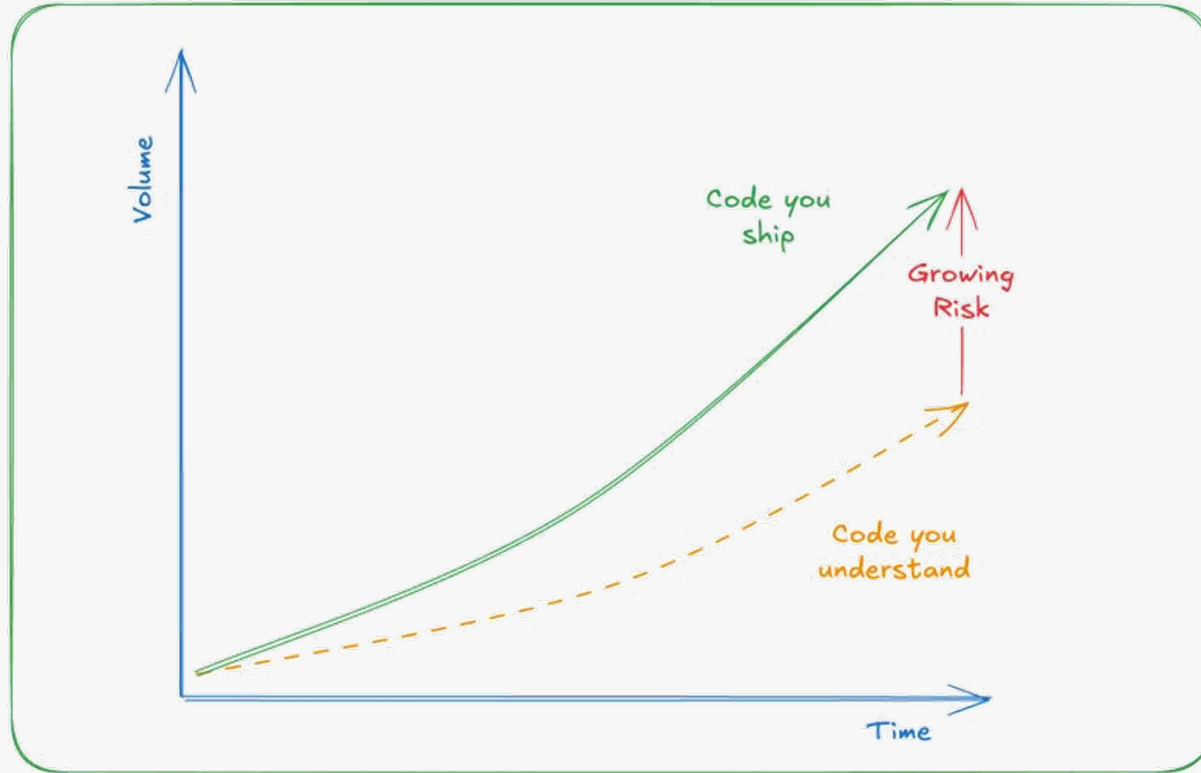
 29 

 Reply  Award  Share ...

 31 more replies

[https://www.reddit.com/r/cursor/comments/1inoryp/cursor\\_fck\\_up\\_my\\_4\\_months\\_of\\_works/](https://www.reddit.com/r/cursor/comments/1inoryp/cursor_fck_up_my_4_months_of_works/)

# Vibe Coding: The GenAI Confidence Gap



*"AI made writing code cheap. Shipping it safely has never been more expensive. That's the paradox most teams are living with right now. **The bottleneck isn't generating code, it's validating and delivering it with confidence.**"*

[https://www.linkedin.com/posts/robzuber\\_ai-made-writing-code-cheap-shipping-it-safely-activity-7377305210483871744-2A1u/](https://www.linkedin.com/posts/robzuber_ai-made-writing-code-cheap-shipping-it-safely-activity-7377305210483871744-2A1u/)

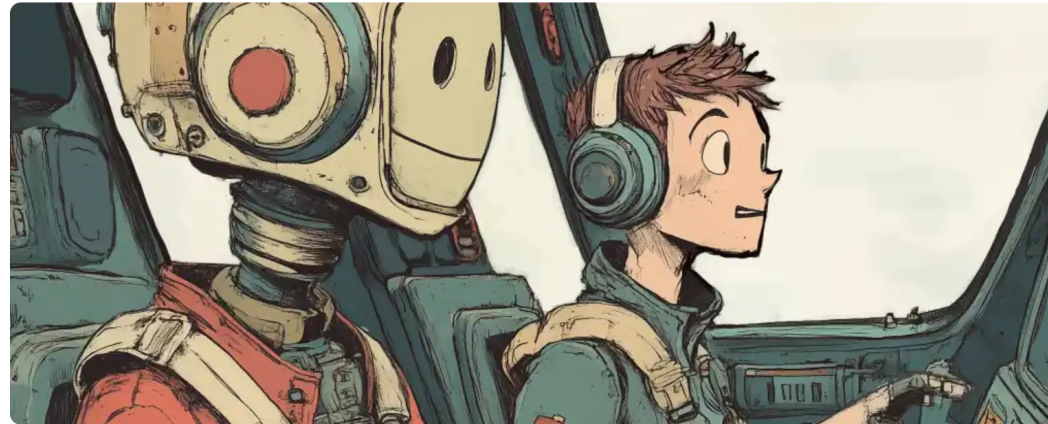


# GenAI Can Introduce “Untypical” Bugs

## Copilot Induced Crash



Klaas van Schelven; January 14 - 5 min read



AI-generated image for an AI-generated bug; as with code, errors are typically different from human ones.

While everyone is talking about how “AI” can help solve bugs, let me share how **LLM-assisted coding gave me 2024’s hardest-to-find bug**.

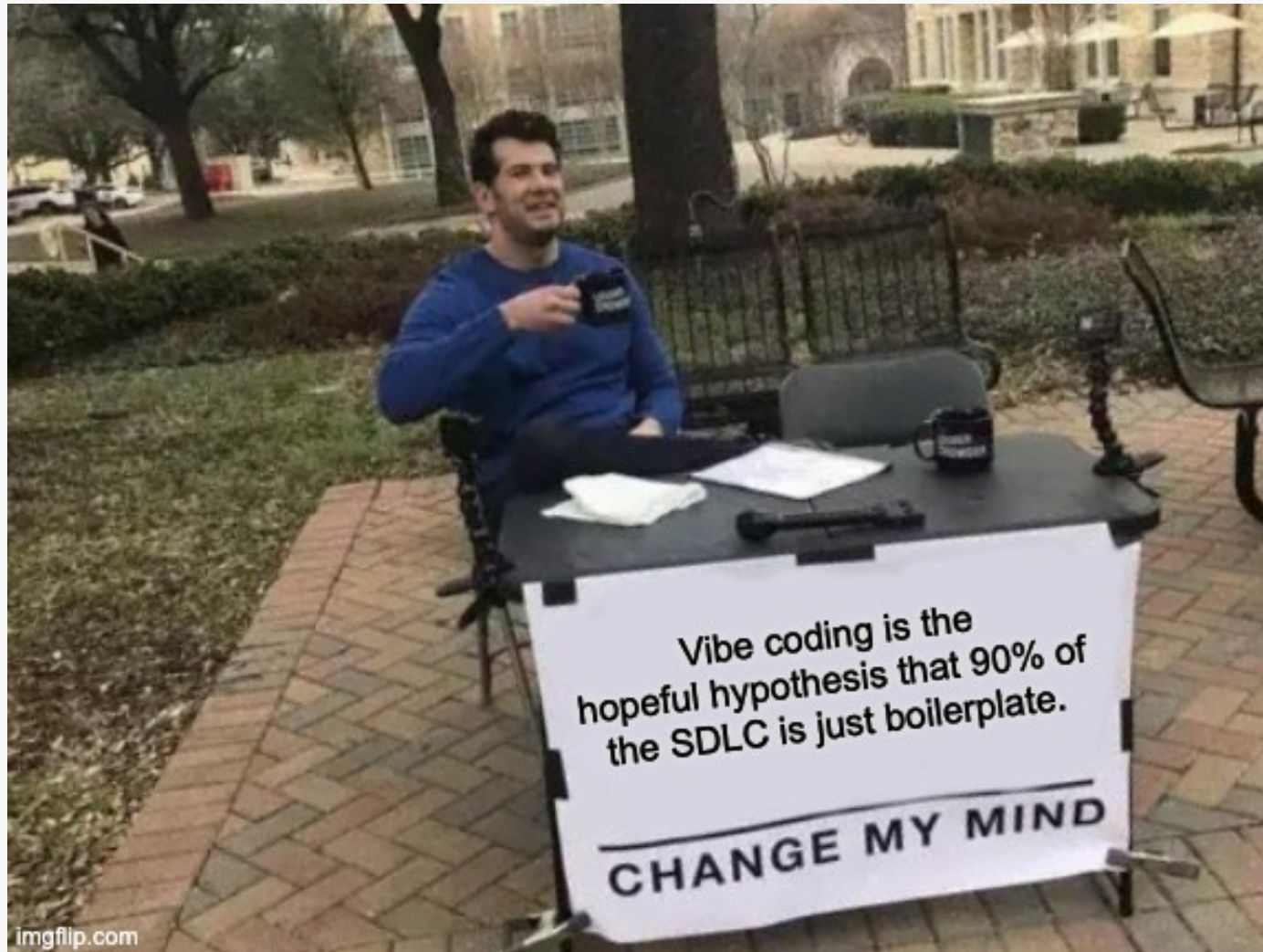
Rather than take you along on my “exciting” debugging journey, I’ll cut to the chase. Here’s the bug that *Microsoft Copilot* introduced for me while I was working on my import statements:

```
from django.test import TestCase as TransactionTestCase
```

<https://www.bugsink.com/blog/copilot-induced-crash/>



# Vibe Coding: My View



<https://imgflip.com/i/9zk6ku>

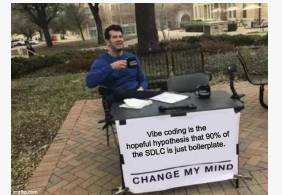
# Four Takeaways of this Talk



One does not simply **measure** software development **efficiency/productivity**.



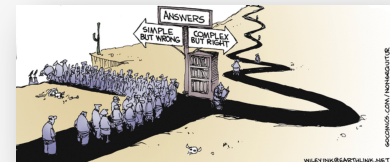
“**Vibe coding**” without following software engineering best practices is **bound to fail**.



“**Trusting AI**” entails much more than just tool **reliance**.



**GenAI** makes good empirical **research harder** rather than easier.



# Working Paper on Trust and GenAI in SE

## Rethinking Trust in AI Assistants for Software Development: A Critical Review

Sebastian Baltes, Timo Speith, Brenda Chiteri, Seyedmoein Mohsenimofidi, Shalini Chakraborty, Daniel Buschek  
University of Bayreuth, Germany

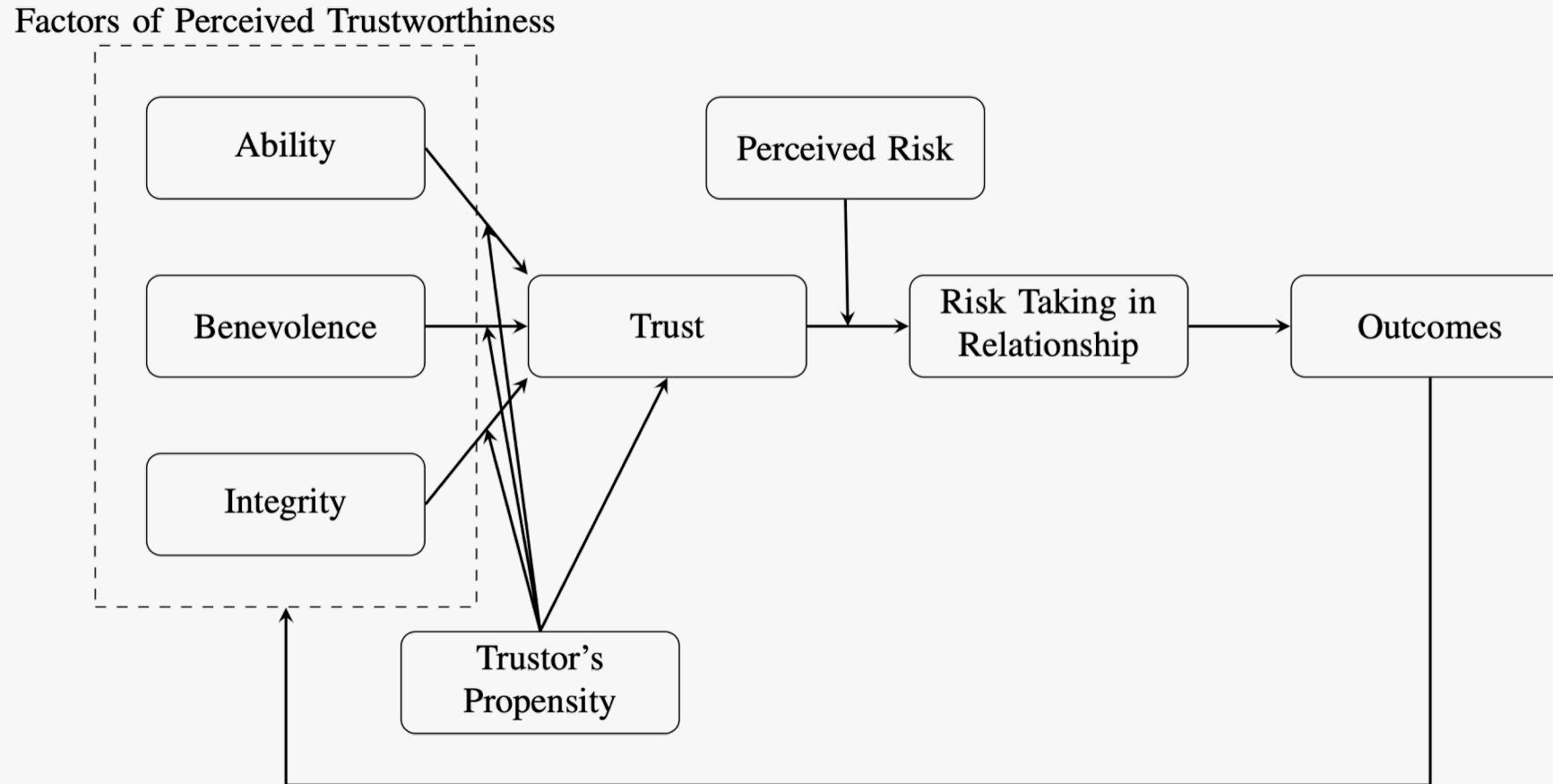
{sebastian.baltes, timo.speith, brenda.chiteri, s.mohsenimofidi, s.chakraborty, daniel.buschek}@uni-bayreuth.de

<https://arxiv.org/pdf/2504.12461>

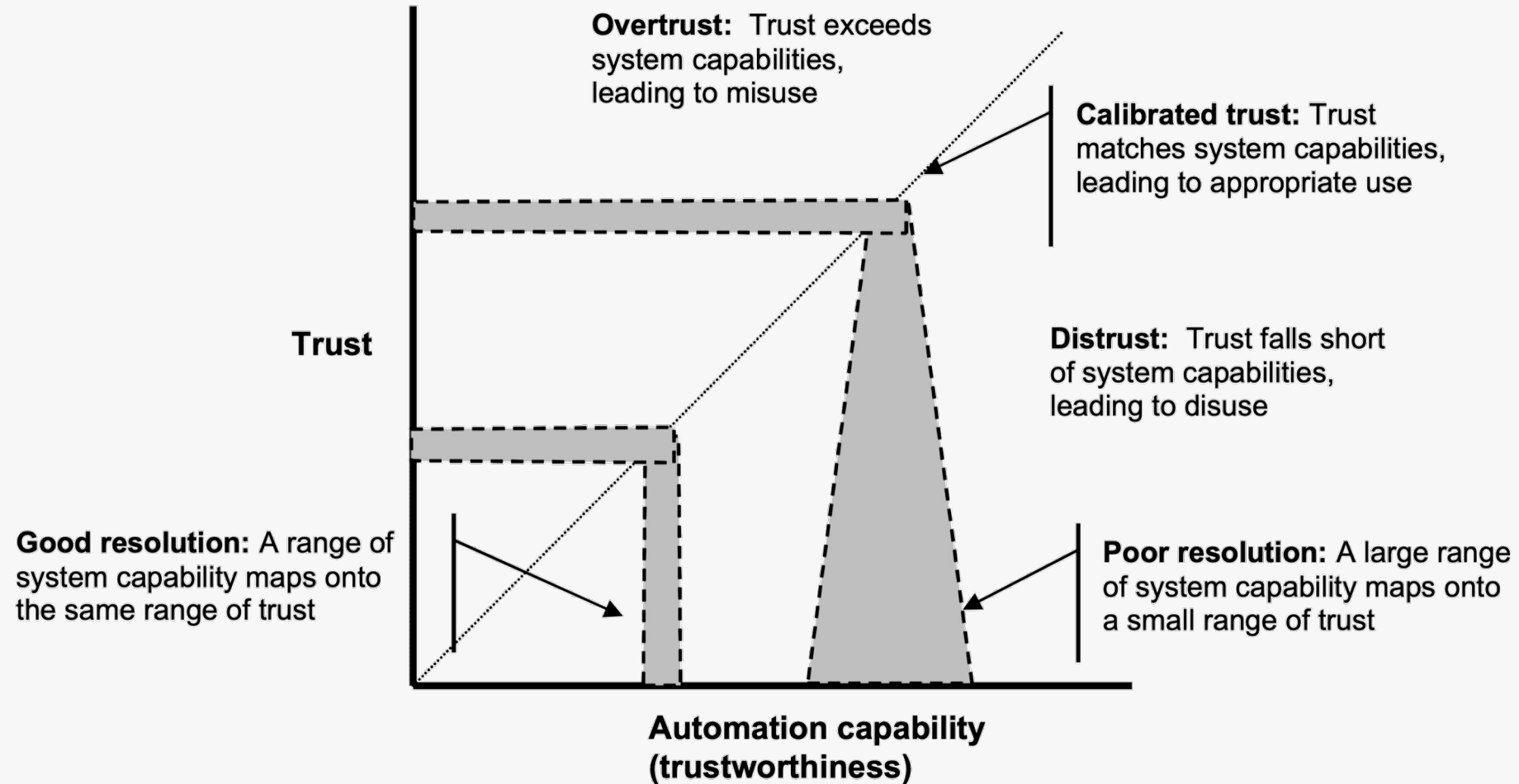
# Scientific Debate on Trust

- **Trust** has been **studied for decades** in fields such as philosophy and psychology, but also in information systems.
- Mayer et al. (1995) defined trust as the *“willingness of a party to be **vulnerable** to the actions of another party based on the expectation that the other will **perform a particular action important** to the trustor, irrespective of the ability to **monitor or control** that other party.”*
- **Applicability** to AI tools actively discussed in philosophy.
- Many important **related concepts**: propensity/disposition to trust (→ initial trust), calibrated/well-grounded trust, under- vs. over-trust, appropriate trust, etc.

# Trust Model by Mayer et al. (1995)



# Trust Model by Lee and See (2004)



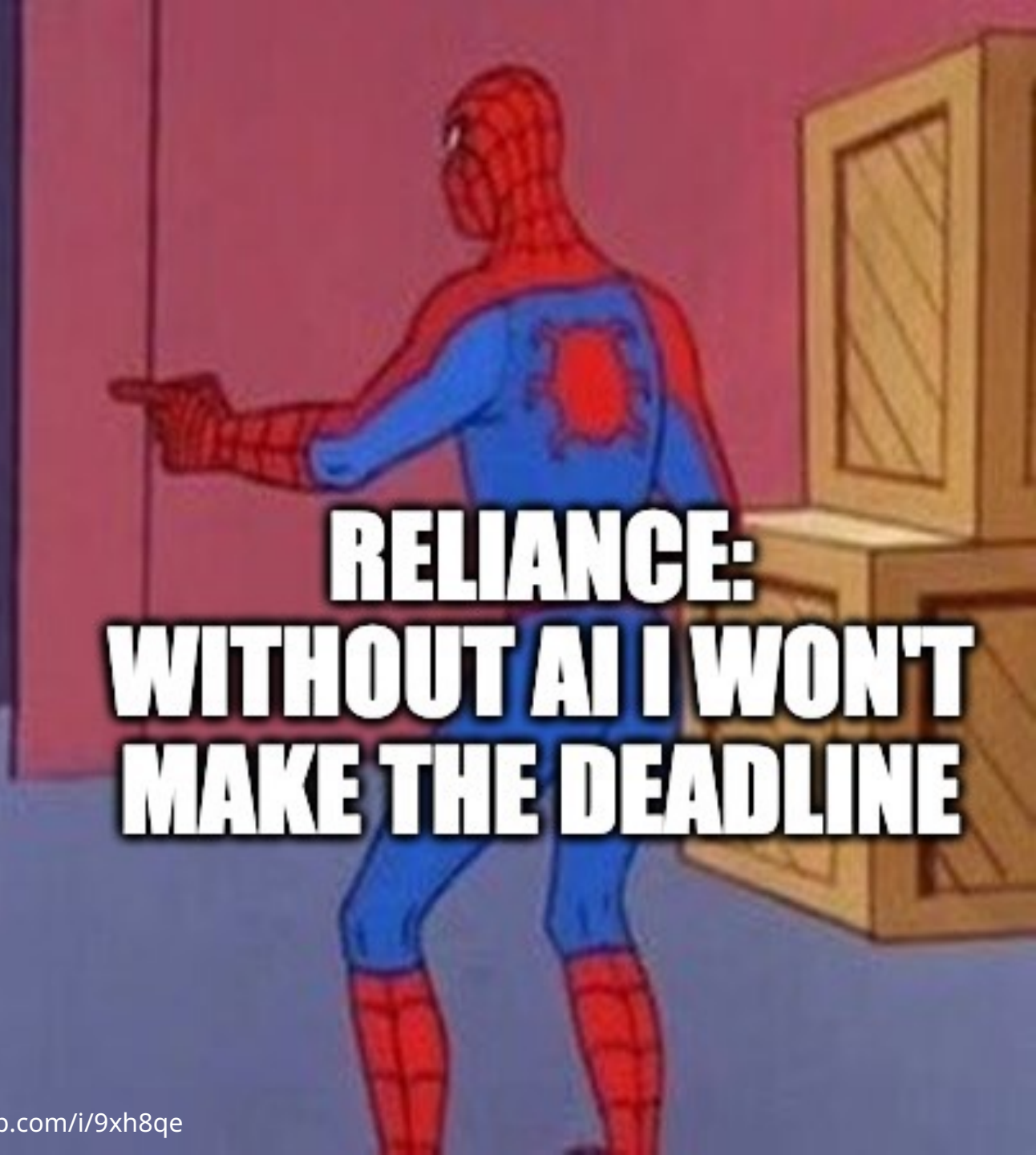
# Meanwhile in Software Engineering...

- Trust is often used **informally**, without a definition, conceptualization or **embedding** in the inter-disciplinary **discourse on trust**.
- GenAI: common to **equate** trust with **artifact acceptance**.  
(acceptance rate for AI-generated software artifacts)
- **Recommendation:**
  - Rather use the term **reliance**, which has a more functional connotation (a dependence on the action of a person/tool to fulfill a need).
  - If you decide to use the term **trust**, look at literature **beyond SE**, refer to **established trust models**, use **established terminology**, and follow the other recommendations in our article 😊





**TRUST:  
CAN AI BETRAY ME?**



**RELIANCE:  
WITHOUT AI I WON'T  
MAKE THE DEADLINE**



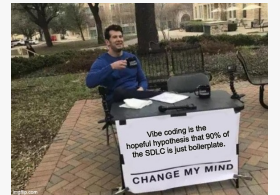
# Four Takeaways of this Talk



One does not simply **measure** software development **efficiency/productivity**.



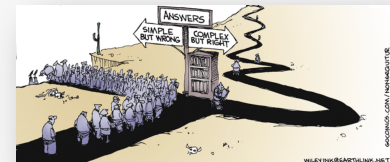
“**Vibe coding**” without following software engineering best practices is **bound to fail**.



“**Trusting AI**” entails much more than just tool **reliance**.



**GenAI** makes good empirical **research harder** rather than easier.



# Coming Back To This...



# Consequences of SE Research Riding the Hype Train...

## Reflections on the Reproducibility of Commercial LLM Performance in Empirical Software Engineering Studies

Florian Angermeir  
angermeir@fortiss.org  
fortiss and Blekinge Institute of  
Technology  
Munich and Karlskrona, Germany  
and Sweden

Andreas Bauer  
andreas.bauer@bth.se  
Blekinge Institute of Technology  
Karlskrona, Sweden

Fabiola Moyón C.  
fabiola.moyon@siemens.com  
Siemens AG  
Munich, Germany

Maximilian Amougou  
amougou@fortiss.org  
fortiss  
Munich, Germany

Matthias Linhuber  
matthias.linhuber@tum.de  
Technical University of Munich  
Munich, Germany

Daniel Mendez  
daniel.mendez@bth.se  
Blekinge Institute of Technology and  
fortiss  
Karlskrona and Munich, Sweden and  
Germany

Mark Kreitz  
mark.kreitz@unibw.de  
University of the Bundeswehr Munich  
Munich, Germany

Davide Fucci  
davide.fucci@bth.se  
Blekinge Institute of Technology  
Karlskrona, Sweden

Tony Gorschek  
tony.gorschek@bth.se  
Blekinge Institute of Technology and  
fortiss  
Karlskrona and Munich, Sweden and  
Germany

“[...] despite a growing interest in LLM-based research in SE, the research field suffers from **fundamental issues with reproducibility** [...]. This **undermines both the scientific value** of current LLM-based publications and the long-term **relevance** of their findings.

<https://arxiv.org/pdf/2510.25506>

# Our Contribution

Contribute via GitHub

## LLM Guidelines for SE

Guidelines for Empirical Studies in Software Engineering involving LLMs.

1. [Declare LLM Usage and Role](#)
2. [Report Model Version, Configuration, and Customizations](#)
3. [Report Tool Architecture beyond Models](#)
4. [Report Prompts, their Development, and Interaction Logs](#)
5. [Use Human Validation for LLM Outputs](#)
6. [Use an Open LLM as a Baseline](#)
7. [Use Suitable Baselines, Benchmarks, and Metrics](#)
8. [Report Limitations and Mitigations](#)



<https://llm-guidelines.org/>

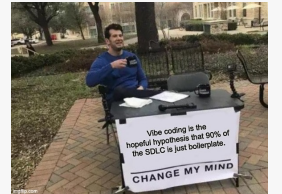
# Four Takeaways of this Talk



One does not simply **measure** software development **efficiency/productivity**.



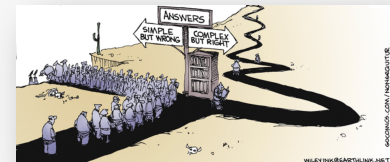
“**Vibe coding**” without following software engineering best practices is **bound to fail**.



“**Trusting AI**” entails much more than just tool **reliance**.



**GenAI** makes good empirical **research harder** rather than easier.





# The Future of Software Engineering?

**Ironies of Automation** (Lisanne Bainbridge, 1983):  
*"[...] severe problems are caused by automating most of the work, while the **human operator is responsible for tasks that can't be automated**. Thus, operators **will not practice skills as part of their ongoing work**. Their work now also includes **exhausting monitoring tasks**. Thus, rather than needing less training, operators **need to be trained more** to be ready for the rare but crucial interventions."*



 empirical-software.engineering

**Prof. Dr. Sebastian Baltes**