

Software Engineering Research Beyond the AI Hype

Prof. Dr. Sebastian Baltes



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

SE@UHD
Software Engineering

Sebastian Baltes

Research Focus: *Software Analytics* (when I talk to practitioners) or *Empirical Software Engineering* (when I talk to academics).

TL;DR: Processing, analyzing, and visualizing software engineering data to monitor, govern, and improve software development processes and tools.

More information: <https://se-research.wtf/>
I registered a new domain, first time I can use it :-)



SE@UHD
Software Engineering



**UNIVERSITÄT
HEIDELBERG**
ZUKUNFT
SEIT 1386

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

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

Industry



Academia



UNIVERSITÄT
BAYREUTH



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Teaching Literature Reviewing for Software Engineering Research



Sebastian Baltes and Paul Ralph

Abstract The goal of this chapter is to support teachers in holistically introducing graduate students to literature reviews, with a particular focus on secondary research. It provides an overview of the overall literature review process and the different types of literature review before diving into guidelines for selecting and conducting different types of literature review. The chapter also provides recommendations for evaluating the quality of existing literature reviews and concludes with a summary of our learning goals and how the chapter supports teachers in addressing them.

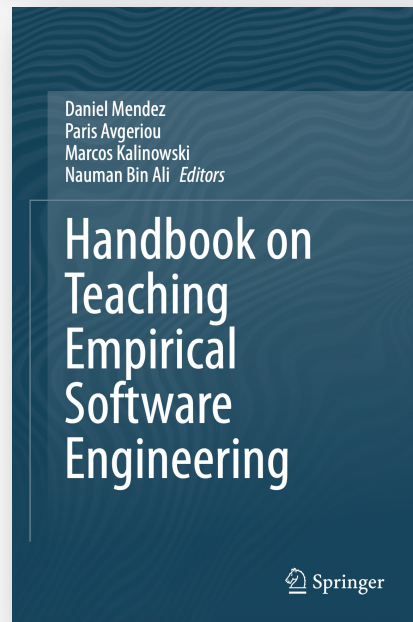


Table 1 The seven types of literature review: categorization

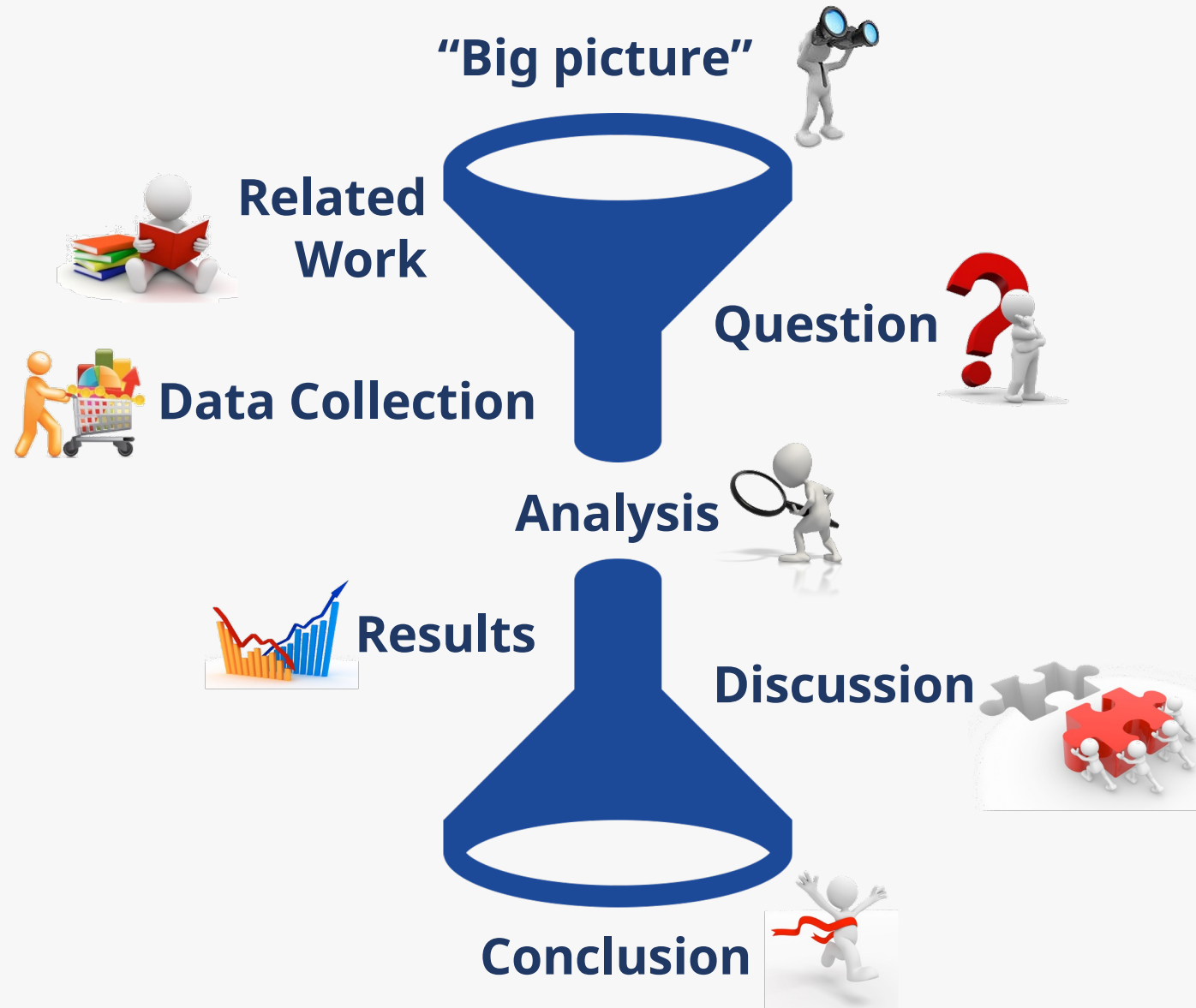
Type	Systematic	Purpose	Primary studies	Analysis
Ad hoc review	No	Discuss	Any	Any
Meta-analysis	Yes	Explain and predict	Quantitative	Quantitative
Meta-synthesis	Yes	Explain	Qualitative	Qualitative
Case survey	Yes	Explain and predict	Qualitative	Quantitative
Critical review	Yes	Prescribe	Any	Any
Scoping review	Yes	Describe	Any	Both
Rapid review	Yes	Explain and Predict	Quantitative	Quantitative

Table 2 The seven types of literature review: comparison

Type	Approach
Ad hoc review	Discuss purposively selected related work.
Meta-analysis	Estimate effect sizes by aggregating results of similar quantitative studies.
Meta-synthesis	Synthesize the findings of numerous studies using qualitative analysis.
Case survey	Test causal hypotheses by aggregating case study results.
Critical review	Defend a position and make recommendations by analyzing a sample of papers.
Scoping review	Describe an area of research and map studies into meaningful categories.
Rapid review	A meta-analytic review that compromises rigor for speed.

<https://arxiv.org/abs/2406.08369>

Funnel of Empirical Research (Papers)



(1) Software Engineering (SE) $\gg$ Programming

(2) AI “solves” Programming $\nRightarrow$ AI “solves” SE

(3) Agentic AI is a Software Systems Innovation

(4) We Need More Solid Empirical Evidence

Current Discourse on AI for SE





Who knows this man?

Boris Cherny

Creator of Claude Code



ANTHROPIC



“Coding Is Solved”

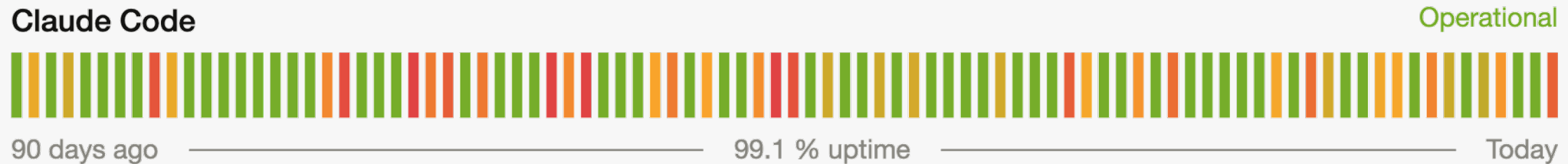
<https://youtu.be/SIGRN8jh2RI>

Well, but Software Engineering isn't

- **Bash**(cd /Users/sebastian/Git/teaching/ise25-26_campus-coffee
mkdir -p build-logic/src/main/kotlin/de/seuhd...)
 - | Error: claude-opus-4-7[1m] is temporarily unavailable, so auto mode cannot determine the safety of Bash right now. Wait briefly and then try this action again. If it keeps failing, continue with other tasks that don't require this action and come back to it later. Note: reading files, searching code, and other read-only operations do not require the classifier and can still be used.
- Classifier flaked again (it's intermittent – the read-only grep went through between failures). Retrying the same operation:
- **Bash**(cd /Users/sebastian/Git/teaching/ise25-26_campus-coffee
mkdir -p build-logic/src/main/kotlin/de/seuhd...)
 - | Error: claude-opus-4-7[1m] is temporarily unavailable, so auto mode cannot determine the safety of Bash right now. Wait briefly and then try this action again. If it keeps failing, continue with other tasks that don't require this action and come back to it later. Note: reading files, searching code, and other read-only operations do not require the classifier and can still be used.
- Classifier still down for write commands. Rather than risk leaving the repo mid-change (the move needs git, which needs Bash), I'll keep it in its current good state (squash `2bfa340`, green) and retry the operation once more – environment hiccups like this usually clear quickly:
- **Bash**(cd /Users/sebastian/Git/teaching/ise25-26_campus-coffee
mkdir -p build-logic/src/main/kotlin/de/seuhd...)
 - | package lines: 6 updated; module id() refs updated: 20
 - | **== build ==**
 - | **BUILD SUCCESSFUL in 45s**
 - | ... +7 lines (ctrl+o to expand)
 - | (timeout 10m)
 - | Allowed by auto mode classifier
- ★ **Schlepping...** (7m 30s · ↓ 17.6k tokens · almost done thinking with max effort)

Well, but Software Engineering isn't

Claude Status



<https://status.claude.com/>

$0.009 \times 90 \text{ days} = 0.81 \text{ days} = 19.44 \text{ hours} = 1,166.4 \text{ minutes}$
→ 12.96 minutes per day*

*Of course, the actual duration and frequency varies (see <https://pulsetic.com/status/claude/>).

Well, but Software Engineering isn't



Boris Cherny ✓

@bcherny

Follow



Coding is just one part of engineering. There's also debugging, operating services, scaling up infrastructure, deciding what to optimize, setting up hardware and capacity, talking to users, product planning, etc. Coding is the easy part, everything else is not yet solved (but is also becoming increasingly automated).

2:04 AM · Jun 9, 2026 · **90.5K** Views

<https://x.com/bcherny/status/2064136590667256229>

*Of course, the actual duration and frequency varies (see <https://pulsetic.com/status/claude/>).

Cursor-Opus agent snuffs out startup's production database

Relax, the data's been recovered. Continue with your vibe coding

 Thomas Claburn

Published Mon 27 Apr 2026 // 21:29 UTC



<https://www.theregister.com/software/2026/04/27/cursor-opus-agent-snuffs-out-startups-production-database/5224442>

An AI-powered coding tool wiped out a software company's database, then apologized for a 'catastrophic failure on my part'

By Beatrice Nolan

Tech Reporter

July 23, 2025, 7:22 AM ET

<https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure/>

AI

Uber's COO says it's getting harder to justify the money spent on AI tokenmaxxing

By Aditi Bharade [+ Follow](#)

<https://www.businessinsider.com/uber-coo-andrew-macdonald-ai-token-spending-harder-justify-2026-5>

Vulnerability Report

CVE-2025-48757: Lovable's Row-Level Security Breakdown Exposes Sensitive Data Across Hundreds of Projects



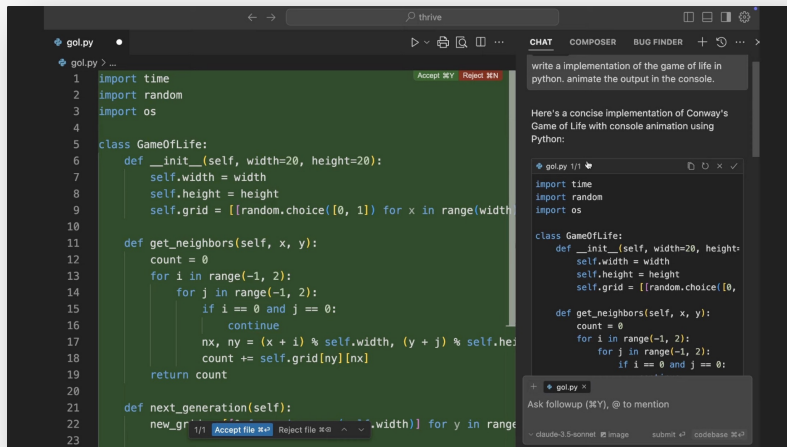
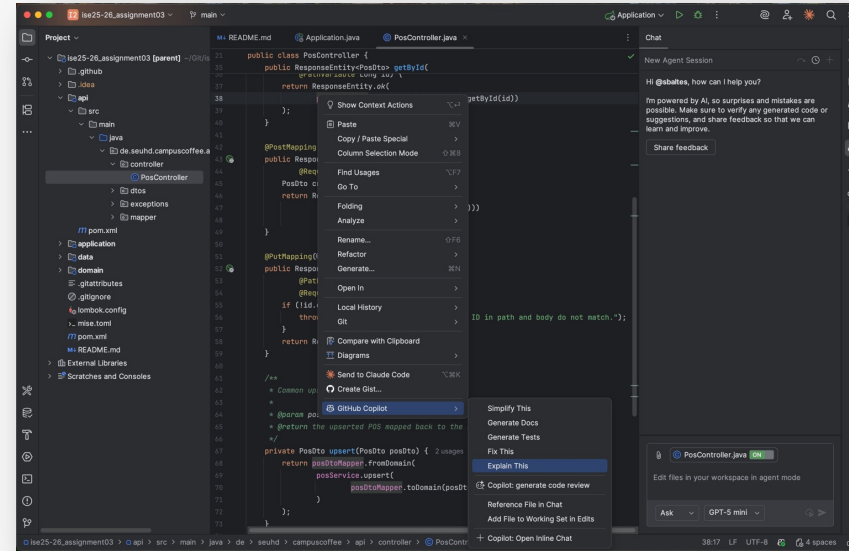
June 10, 2025 3 minutes read 0

<https://securityonline.info/cve-2025-48757-lovables-row-level-security-breakdown-exposes-sensitive-data-across-hundreds-of-projects/>

Generative AI Tools for Software Development

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

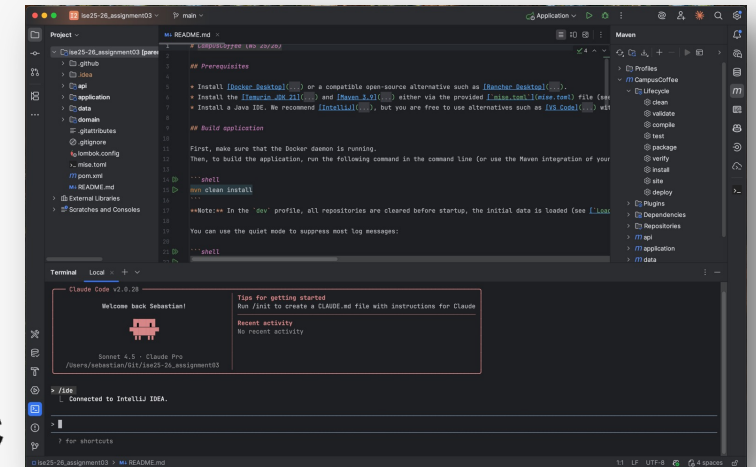
 **GitHub Copilot**



GenAI-
centric **IDEs**



“Agentic” AI
assistants



What is Software Engineering?

Let's start with what it isn't



Who knows this man?

Andrej Karpathy
Co-founder of OpenAI

OpenAI ANTHROPIC

Vibe Coding



Andrej Karpathy 
@karpathy



There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

12:17 AM · Feb 3, 2025 · 4.9M Views



1.3K



4.9K



29K



15K



*"There's a new kind of coding I call **vibe coding** [...] where you [...] forget that the code even exists."*

*"I just see stuff, say stuff, run stuff, and copy paste stuff, and it **mostly works**."*

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

Vibe Coding



*"There's a new kind of coding I call **vibe coding** [...] where you [...] forget that the code even exists."*

*"I just see stuff, say stuff, run stuff, and copy paste stuff, and it **mostly works**."*

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

Where Vibe Coding Works: Small Projects

Wunschkennzeichen

1

Antragsteller

Persönliche Daten, Datenschutz

Antragsteller

Name

Datenschutz

Es ist mir bewusst, dass die in den folgenden Schritten erfassten Daten automatisiert verarbeitet werden. Ich stimme der elektronischen Verarbeitung der freiwillig abgegebenen Daten ausdrücklich zu.

☐ Ja, ich stimme zu

ABBRECHEN

WEITER

2

Suche

Kennzeichenauswahl

3

4

Abschluss

Bestätigung

The screenshot shows a GitHub repository named 'license-plate-checker' by user 'sbaltes'. The repository is public and has 1 branch and 0 tags. The file list includes .gitignore, CLAUDE.md, README.md, check_plates.py, requirements.txt, and test_check_plates.py. The README file is open, showing the project's purpose: 'License Plate Availability Checker'. It describes the tool as a GWT (Google Web Toolkit) SPA with obfuscation, using Playwright to drive a headless Chromium browser. The terminal window shows the application's output for the district 'MTK'. It lists 31 combinations of license plate numbers (MTK-SN 1 to MTK-SN 31) and indicates that all are 'TAKEN'.

```
license-plate-checker Public
main 1 Branch 0 Tags
sbaltes Add initial version 526f024 · 3 months ago 1 Commit
.gitignore Add initial version 3 months ago
CLAUDE.md Add initial version 3 months ago
README.md Add initial version 3 months ago
check_plates.py Add initial version 3 months ago
requirements.txt Add initial version 3 months ago
test_check_plates.py Add initial version 3 months ago
README
License Plate Availability Checker
Batch-checks desired license plate ("Wunschkennzeichen") using browser automation.
The portal is a GWT (Google Web Toolkit) SPA with obfuscation.
This tool uses Playwright to drive a headless Chromium browser.
District [MTK]:
Letter combinations (1-2 chars, comma-separated, e.g. AB,CD,XY): SN
Number range (e.g. 1-99 or 1,5,10-50): 1-99
Will check 99 combination(s) for district MTK (Main-Taunus-Kreis).
Navigating to https://kfzonline.ekom21.de/kfzonline.public/start.html?oe=00.00.06.436000 ...
District prefix: MTK
[1/99] MTK-SN 1 -> TAKEN x
[2/99] MTK-SN 2 -> TAKEN x
[3/99] MTK-SN 3 -> TAKEN x
[4/99] MTK-SN 4 -> TAKEN x
[5/99] MTK-SN 5 -> TAKEN x
[6/99] MTK-SN 6 -> TAKEN x
[7/99] MTK-SN 7 -> TAKEN x
[8/99] MTK-SN 8 -> TAKEN x
[9/99] MTK-SN 9 -> TAKEN x
[10/99] MTK-SN 10 -> TAKEN x
[11/99] MTK-SN 11 -> TAKEN x
[12/99] MTK-SN 12 -> TAKEN x
[13/99] MTK-SN 13 -> TAKEN x
[14/99] MTK-SN 14 -> TAKEN x
[15/99] MTK-SN 15 -> TAKEN x
[16/99] MTK-SN 16 -> TAKEN x
[17/99] MTK-SN 17 -> TAKEN x
[18/99] MTK-SN 18 -> TAKEN x
[19/99] MTK-SN 19 -> TAKEN x
[20/99] MTK-SN 20 -> TAKEN x
[21/99] MTK-SN 21 -> TAKEN x
[22/99] MTK-SN 22 -> TAKEN x
[23/99] MTK-SN 23 -> TAKEN x
[24/99] MTK-SN 24 -> TAKEN x
[25/99] MTK-SN 25 -> TAKEN x
[26/99] MTK-SN 26 -> TAKEN x
[27/99] MTK-SN 27 -> TAKEN x
[28/99] MTK-SN 28 -> TAKEN x
[29/99] MTK-SN 29 -> TAKEN x
[30/99] MTK-SN 30 -> TAKEN x
[31/99] MTK-SN 31 -> TAKEN x
```

Where Vibe Coding Works: Small Projects

The image displays three overlapping screenshots of GitHub repository pages, illustrating small projects where 'vibe coding' is applicable. The repositories are:

- caldav-utils** (Public): A repository by sbaltes and claude. The file list includes `.gitignore`, `LICENSE`, `README.md`, `caldav_utils.py`, and `requirements.txt`. The README section is partially visible, showing the project's purpose: "Purge, deduplicate, ... Works with standard ...".
- imap-dedup** (Public): A repository by sbaltes. The file list includes `.gitignore`, `LICENSE`, `README.md`, `imap_dedup.py`, `offlineimaprc`, `sync-mailbox.sh`, and `test_imap_dedup.py`. The README section is partially visible, showing the project's purpose: "Deduplicate emails in a ...".
- openreview-downloader** (Public): A repository by sbaltes and claude. The file list includes `.gitignore`, `LICENSE`, `README.md`, `download_reviews.py`, and `requirements.txt`. The commit history shows several updates, including adding initial versions of scripts and licenses, and adding flags for comments and output format. The README section is visible, showing the project's purpose: "OpenReview Review Downloader. Download reviews, metareviews, decisions, and optionally comments from [OpenReview](#) for a given paper."

Where It Doesn't Work: Publicly Deployed Software

ANDY GREENBERG

SECURITY MAY 7, 2026 7:00 AM

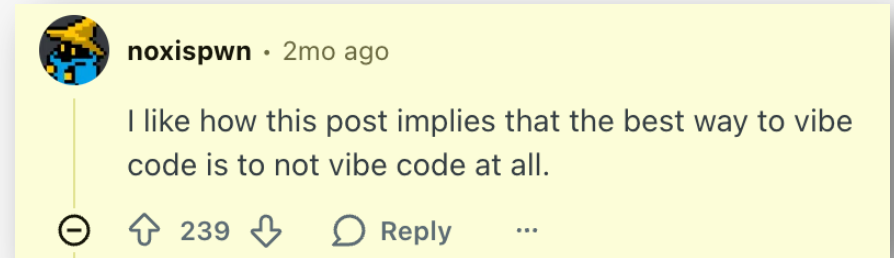
Thousands of Vibe-Coded Apps Expose Corporate and Personal Data on the Open Web

Companies like Lovable, Base44, Replit, and Netlify use AI to let anyone build a web app in seconds—and in thousands of cases, spill highly sensitive data onto the public internet.

<https://www.wired.com/story/thousands-of-vibe-coded-apps-expose-corporate-and-personal-data-on-the-open-web/>

“Vibe Coding” at FAANG

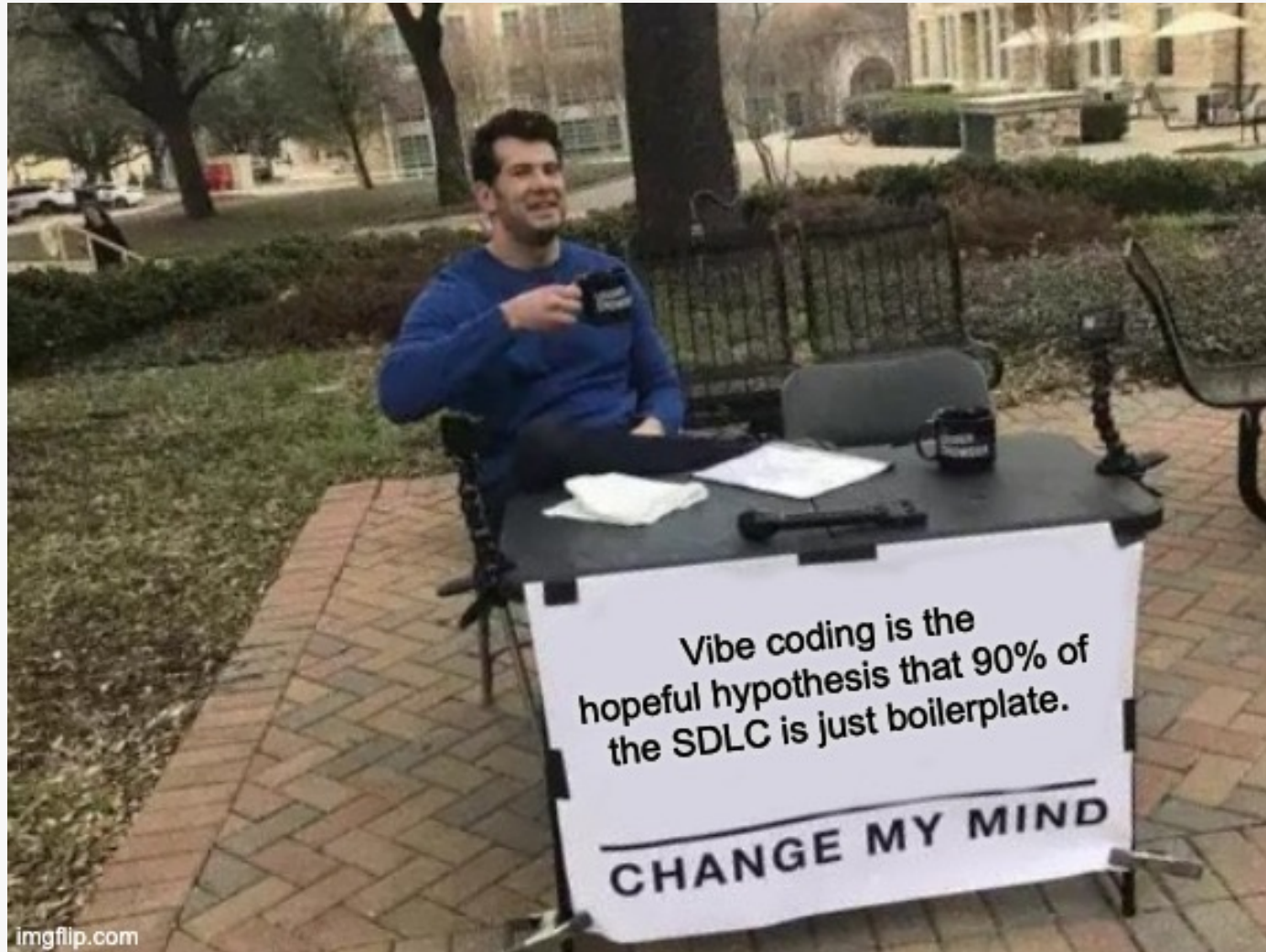
1. [...] start with a **technical design document**.
2. **Design review** before [...] development [...].
3. If you pass review, [...] **more documentation on each subsystem**.
4. **Backlog development** and **sprint planning**.
5. **Software development** [...] We use **test-driven development**, so I have the AI coding agent write the tests first for the feature I’m going to build.
6. **Code** [...] **review**. We have a two-dev approval process before code can get merged into main. AI is [...] assisting with the review.
7. **Test in staging**. If staging is good to go, we push to **prod**.



* Facebook (Meta), Apple, Amazon, Netflix, and Google (Alphabet)

[https://www.reddit.com/r/vibecoding/comments/1myakhd/how we vibe code at a faang/](https://www.reddit.com/r/vibecoding/comments/1myakhd/how_we_vibe_code_at_a_faang/)

Vibe Coding and the SDLC



SDLC: Software Development Life Cycle

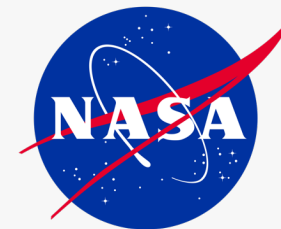
What is the Software Development Life Cycle (SDLC)?



Who knows this woman?

Margaret Hamilton

Led Software Development of
NASA's Apollo Guidance Computer



Origin of the Term “Software Engineering”



Margaret Hamilton
(1965-1969)



Anthony G. Oettinger
(1966)



NATO Software Engineering Conference
(1968/69)

“Software Engineering” as a Discipline (~1968)



Call to action:
“We know how to build **hardware**, now we must study how to build **software**.”

“Software Crisis”:

- Software was **inefficient** and difficult to **maintain**.
- It often did **not meet** the specified **requirements**.
- It was sometimes **not delivered**.

SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

1968 NATO SE Conference Report: The SDLC

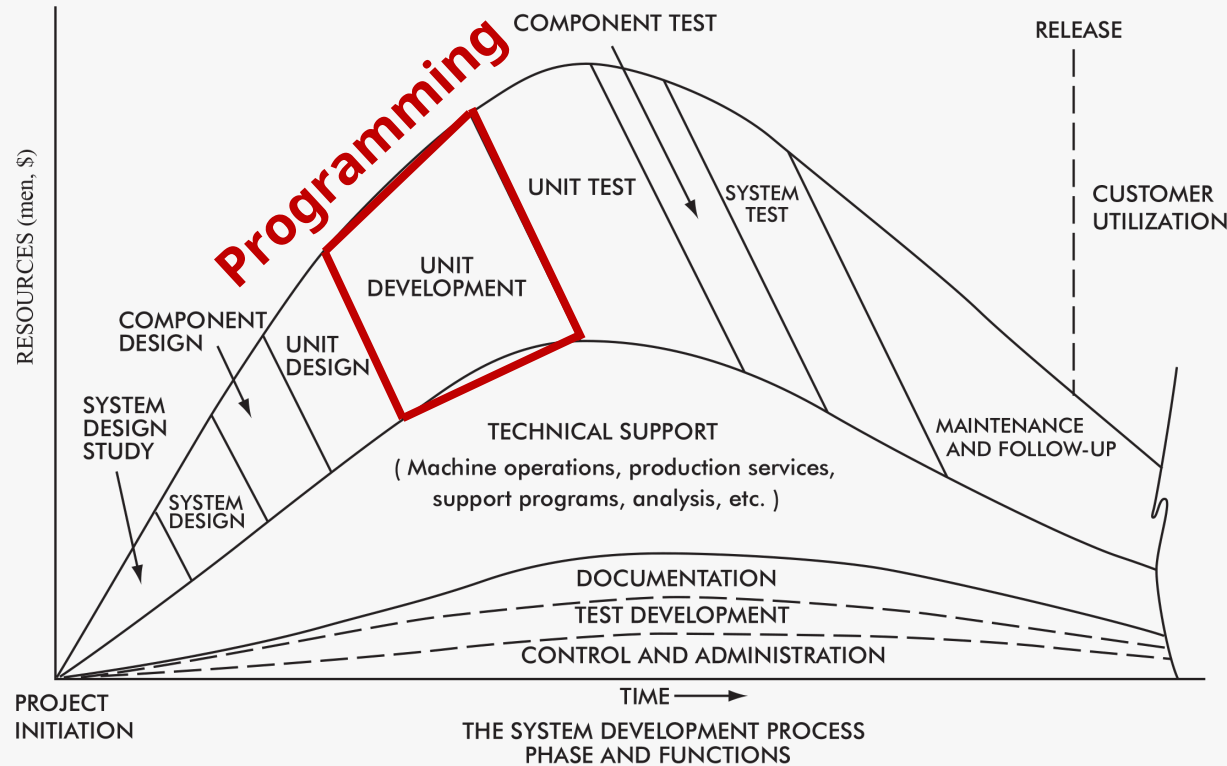


Figure 1. From Nash: Some problems in the production of large-scale software systems.

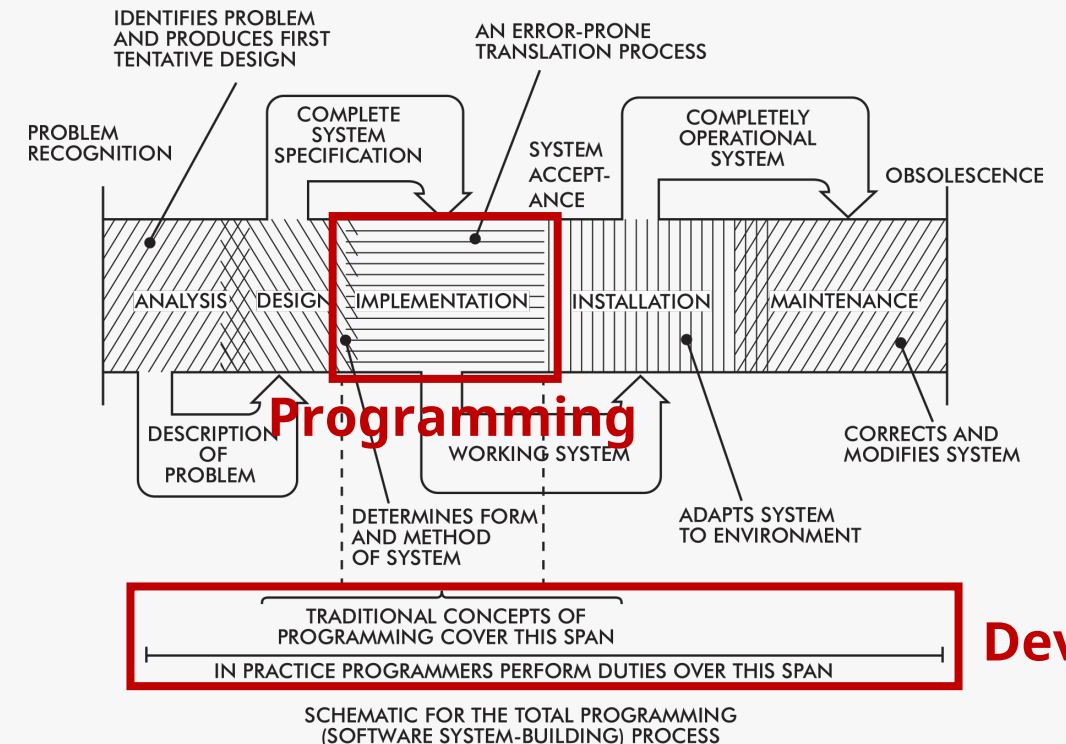
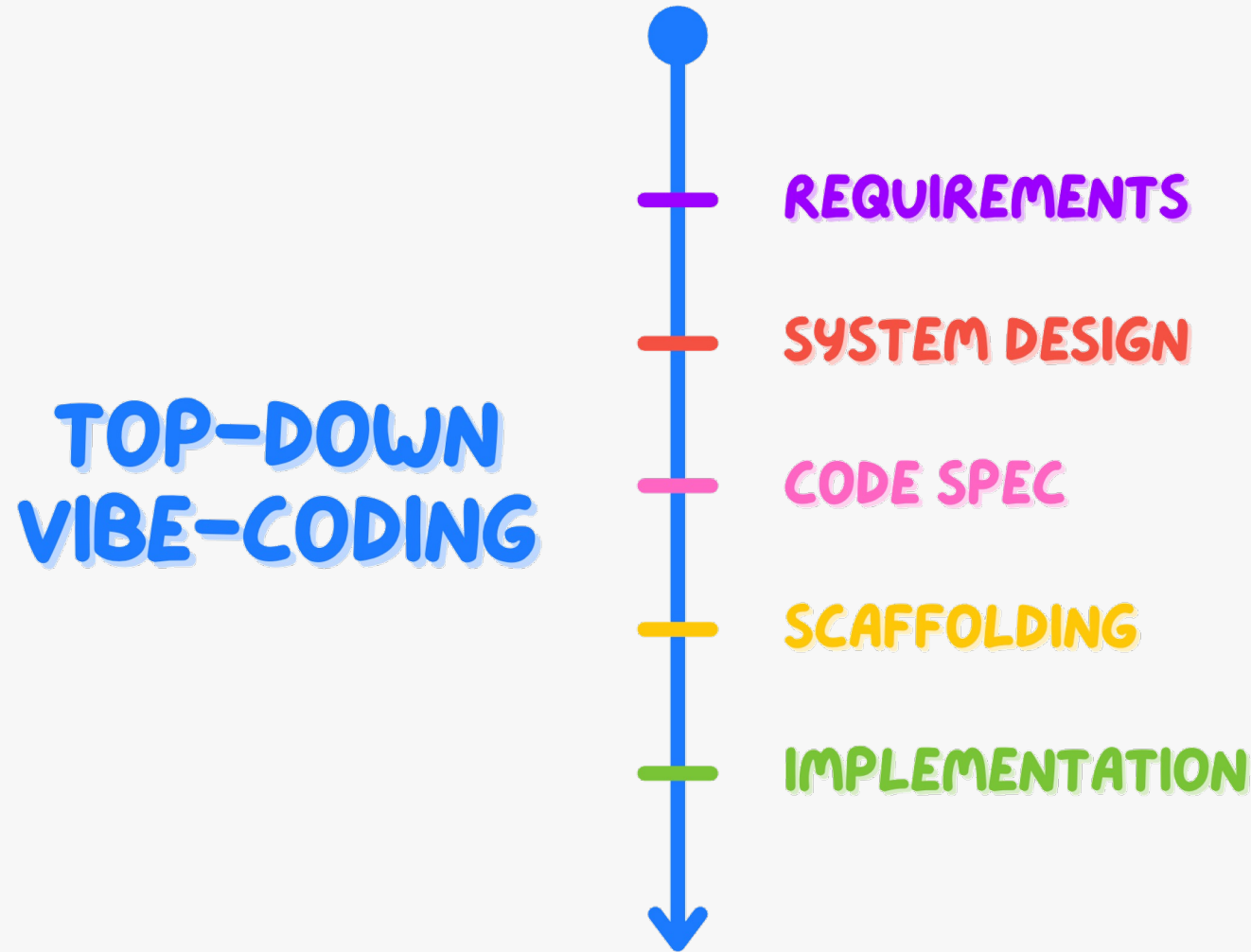


Figure 2. From Selig: Documentation for service and users. Originally due to Constantine.

The SDLC, aka “Top-Down Vibe Coding”



<https://refactoring.fm/p/vibe-coding-workflows>

The SDLC and Agentic AI



<https://github.com/addyosmani/agent-skills>

Software Engineering: Definition

“Software engineering is the **systematic application** of scientific and technological **knowledge, methods, and experience...**

...to the design, implementation, testing, maintenance, operation, and documentation of **software...**

...with a particular focus on **automation.**”

(adapted from ISO/IEC 2382:2015)

Software Engineering: Definition

"Software engineering and technological innovation"

...to the design,
and documenta

...with a particul

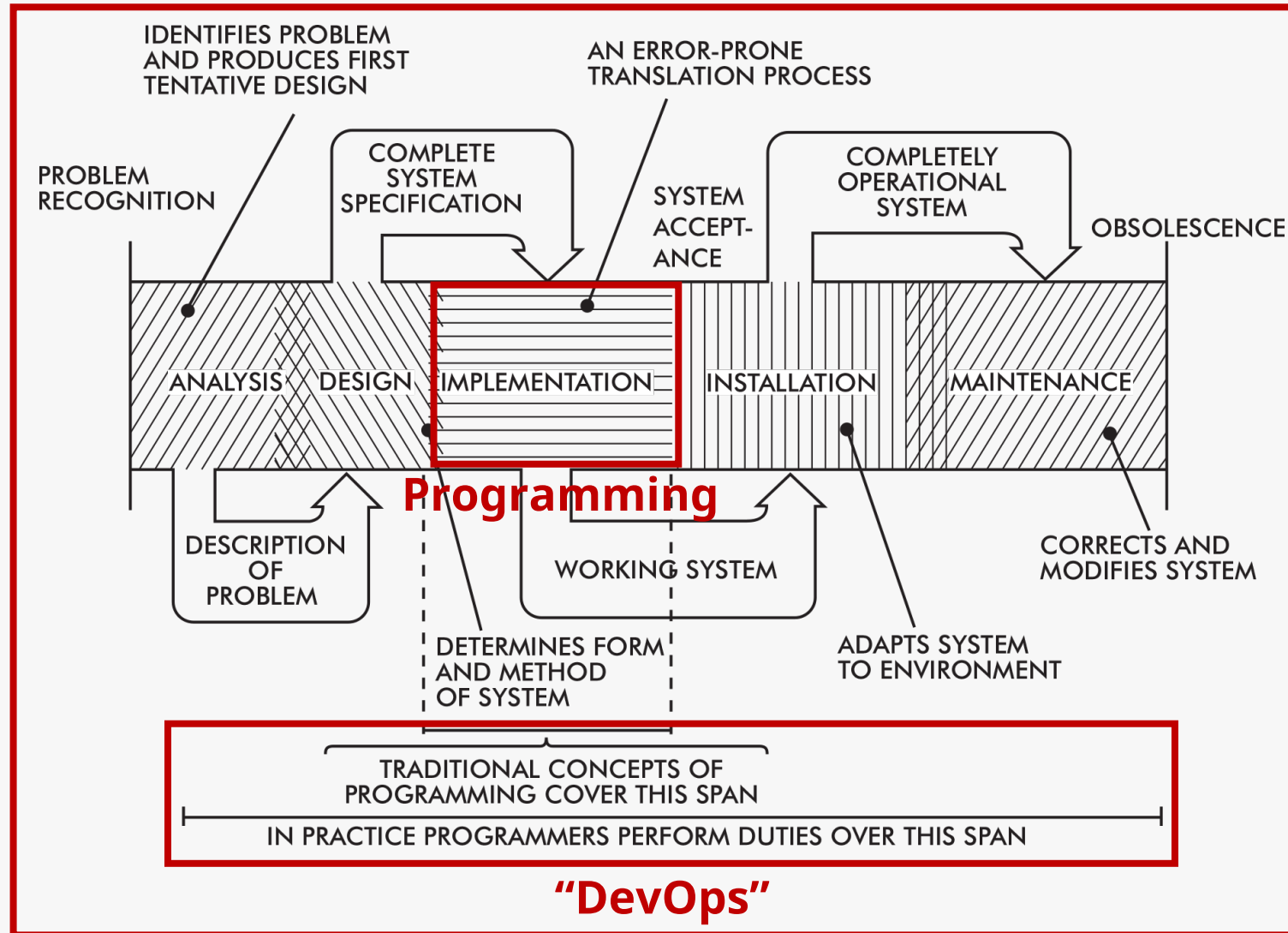


ion of scientific perience...

ance, operation,

from ISO/IEC 2382:2015)

SE Has Never Been Just About Coding!



Software Engineering

(1) Software Engineering (SE) $\gg$ Programming



(2) AI “solves” Programming $\nRightarrow$ AI “solves” SE



(3) Agentic AI is a Software Systems Innovation

(4) We Need More Solid Empirical Evidence

Myths on Software Engineering and GenAI

Myth 1: Developers Spend Most of Their Time Writing Code

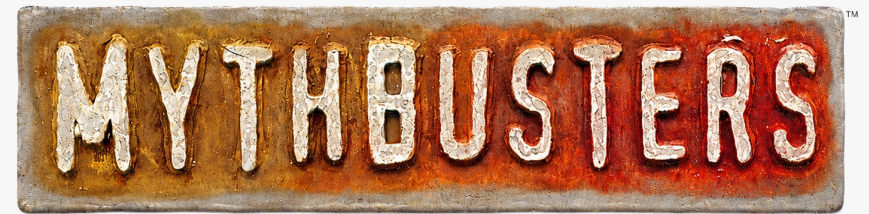
*“A study of more than 450 engineers at Microsoft in 2025 showed **developers spend only 14 percent of their time writing code**, reflecting what has been found in studies over the years.”*

RESEARCH-ARTICLE | 26 May 2026

Eight Myths on Software Engineering and GenAI

Examining the most common misconceptions

Jenna Butler, Brian Houck, Margaret-Anne Storey, Travis Lowdermilk, Steven Clarke, and Emerson Murphy-Hill



<https://spawn-queue.acm.org/doi/10.1145/3807963>



Yes, you can measure software developer productivity

August 17, 2023 | Article

B.S. Producers

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

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

No Single Metric Captures Productivity

Ciera Jaspan, Google, USA

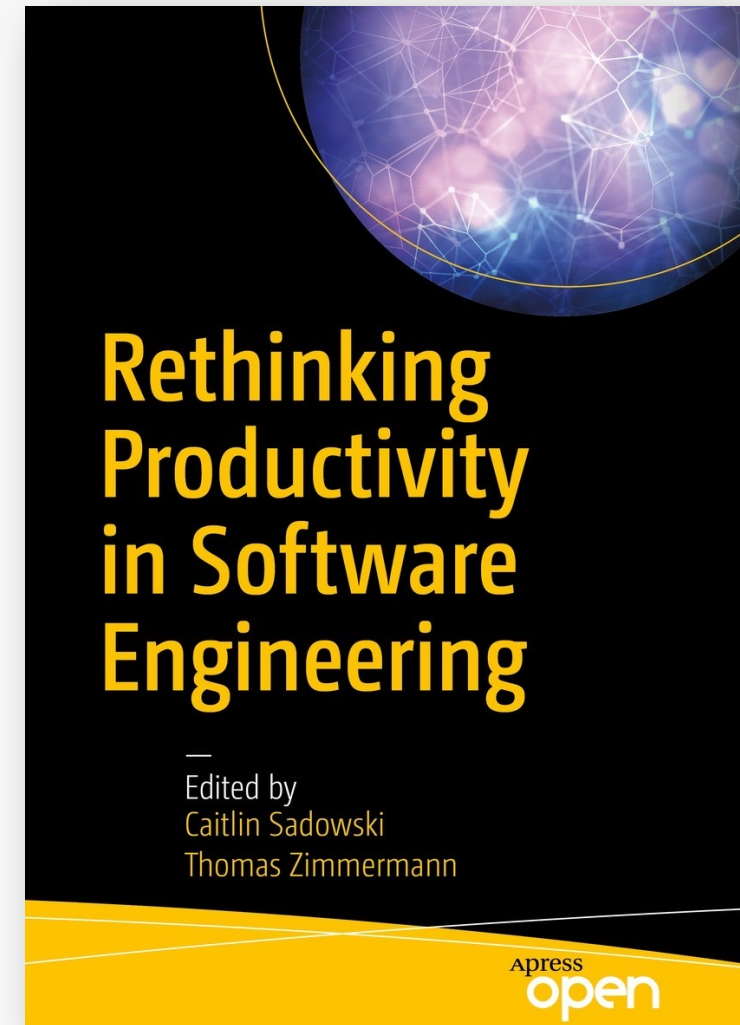
Caitlin Sadowski, Google, USA

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

—Bill Gates

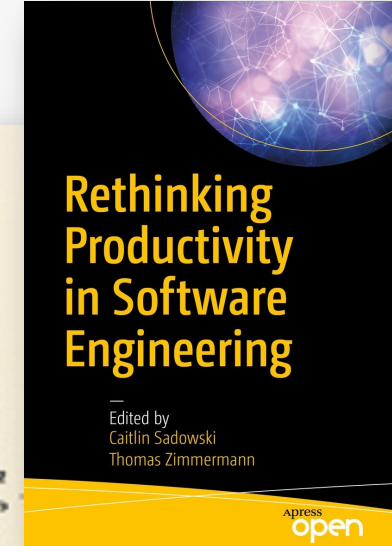
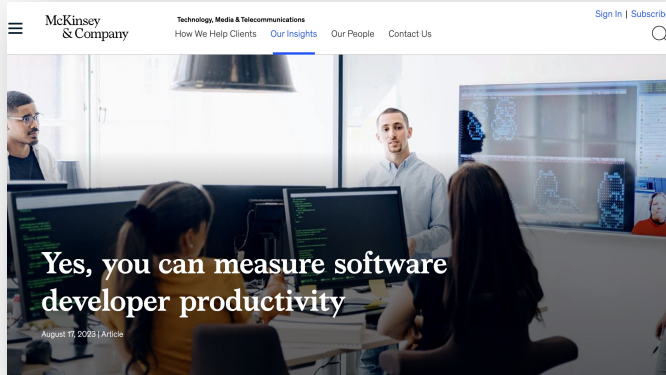
Why We Should Not Measure Productivity

Amy J. Ko, University of Washington, USA



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

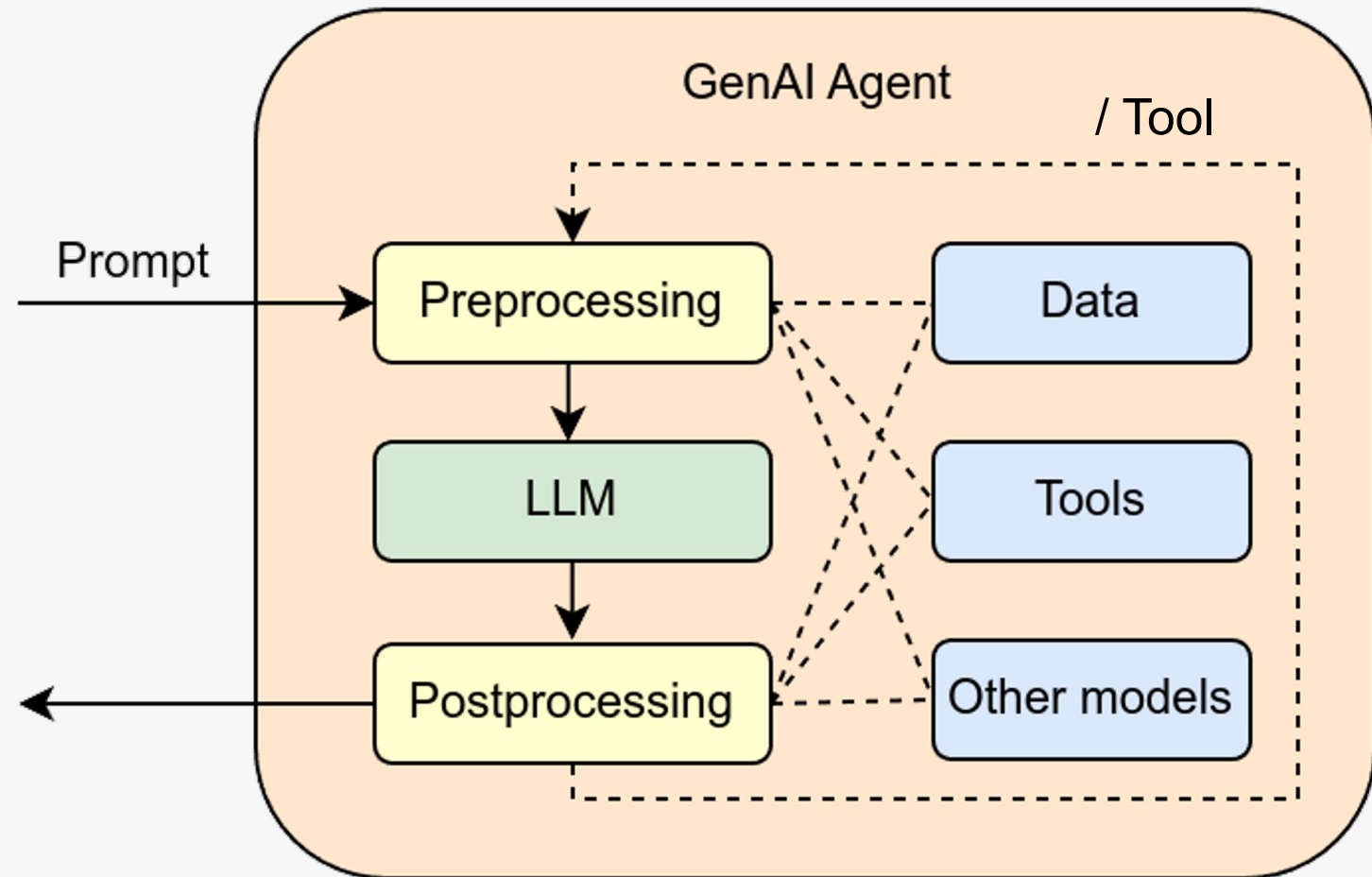
Public Discourse about SE



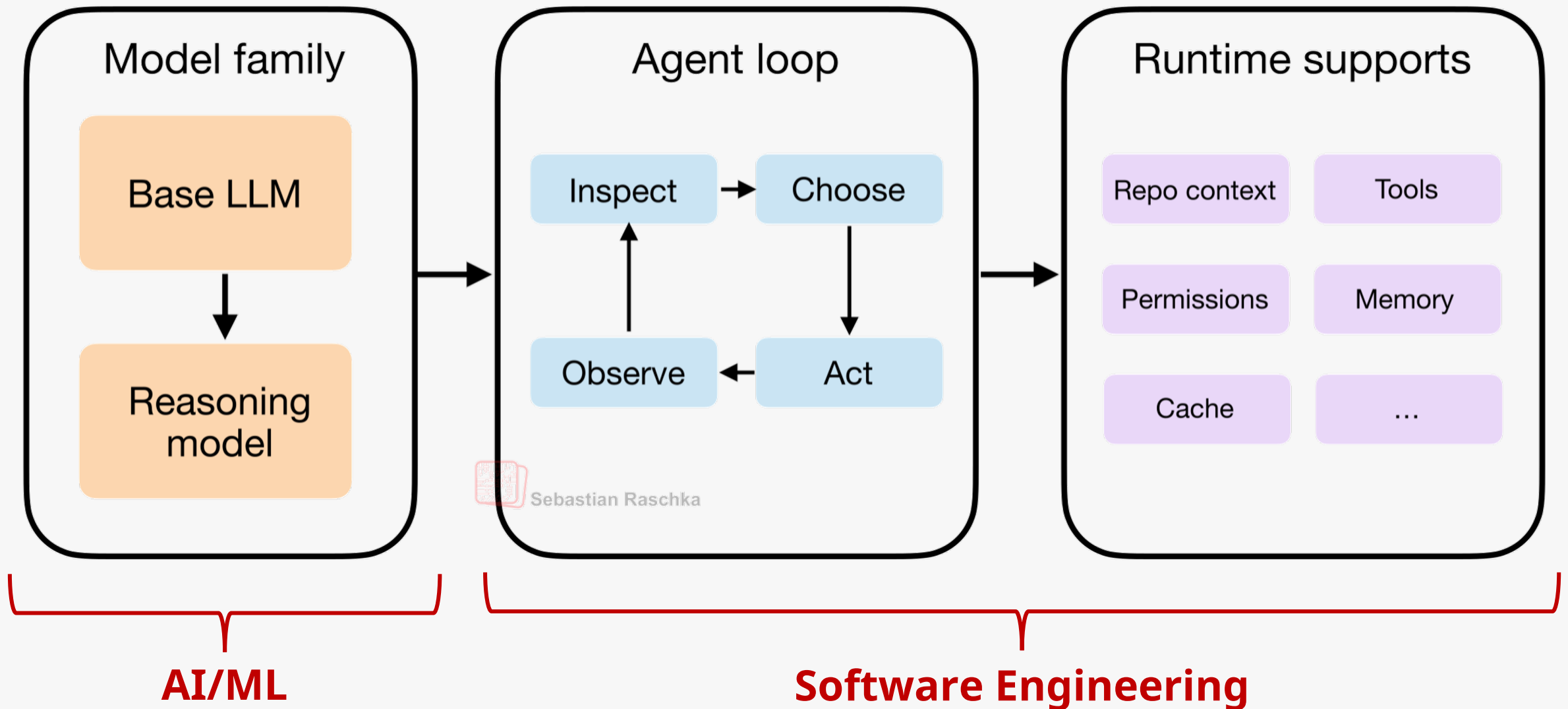
Agentic AI Assistants

Generative Artificial Intelligence (GenAI) Tools

The quality of the **context** provided as part of the prompts is crucial.



Agentic Coding Assistants



<https://magazine.sebastianraschka.com/p/components-of-a-coding-agent>

D: Deterministic
ND: Non-deterministic



TL;DR

We've moved from model innovation to systems innovation

(1) Software Engineering (SE) $\gg$ Programming



(2) AI “solves” Programming $\nRightarrow$ AI “solves” SE



(3) Agentic AI is a Software Systems Innovation



(4) We Need More Solid Empirical Evidence

Benchmarking AI Tools for SE

Benchmarking AI Tools

- Benchmarks are a central aspect of **vendors' marketing**.
- Most **benchmarks are open**, but the **models are closed**.
("Open" usually means open weight, not open training data.)
- **Benchmark contamination** is one of many problems.
- Another problem is that benchmarks **rarely define the construct they intend to measure** (functional correctness, security, reliability, etc.) and why the chosen **operationalization** is valid.

The Leaderboard Illusion

Shivalika Singh^{*1}, Yiyang Nan¹, Alex Wang², Daniel D'souza¹,
Sayash Kapoor³, Ahmet Üstün¹, Sanmi Koyejo⁴, Yuntian Deng⁵,
Shayne Longpre⁶, Noah A. Smith^{7,8}, Beyza Ermiş¹,
Marzieh Fadaee¹, and Sara Hooker¹

¹Cohere Labs, ²Cohere, ³Princeton University, ⁴Stanford University, ⁵University of Waterloo,
⁶Massachusetts Institute of Technology, ⁷Allen Institute for Artificial Intelligence, ⁸University of
Washington

<https://arxiv.org/pdf/2504.20879>

Contamination Report for Multilingual Benchmarks

Sanchit Ahuja*
Microsoft Research

Varun Gumma*
Microsoft Research

Sunayana Sitaram
Microsoft Research

<https://arxiv.org/pdf/2410.16186>

Benchmarks: HumanEval

Prompt:

```
def is_palindrome(text: str):  
    """  
    Checks if given string is a palindrome  
    >>> is_palindrome('')  
    True  
    >>> is_palindrome('aba')  
    True  
    [...]
```

Canonical Solution:

```
def is_palindrome(text: str):  
    for i in range(len(text)):   
        if text[i] != text[len(text)-1-i]:  
            return False  
    return True
```

Tests:

```
assert is_palindrome('') == True  
assert is_palindrome('aba') == True  
assert is_palindrome('aaaaa') == True  
assert is_palindrome('zabcd') == False  
assert is_palindrome('xywyx') == True  
assert is_palindrome('xywyz') == False  
assert is_palindrome('xywzx') == False
```

<https://github.com/openai/human-eval/>

Benchmarks: HumanEval

Check an LLM-generated solution by validating that the generated code is **executable** and **passes the tests**. Usually, several (n) candidate solutions are generated.

$$\text{pass}@k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

pass@k estimates the probability that at least one of k generated candidate solutions for a fixed task passes the benchmark's tests.

where:

- n is the total number of samples generated per prompt (with $n \geq k$),
- c is the number of correct samples among n , and
- k is the number of attempts considered, drawn from the n generated samples without replacement.

Benchmarks: HumanEval

Frontier intelligence at 2x the speed

Claude 3.5 Sonnet sets new industry benchmarks for graduate-level reasoning (GPQA), undergraduate-level knowledge (MMLU), and coding proficiency (HumanEval). It shows marked improvement in grasping nuance, humor, and complex instructions, and is exceptional at writing high-quality content with a natural, relatable tone.

- Just Python.
- Just pure functions.
- No modules.
- No existing codebase.
- No debugging.
- No non-functional properties.

ANTHROPIC
Marketing in June 2024

pass@k estimates the probability that at least one of k generated candidate solutions for a fixed task passes the benchmark's tests.

Construct

Operationalization

Benchmarks: HumanEval

Frontier intelligence at 2x the speed

Claude 3.5 Sonnet sets new industry benchmarks for graduate-level reasoning

coding proficiency

g nuance, humor, and
-quality content with a

**Construct
validity?!**

**pass@k estimates the probability
that at least one of k generated
candidate solutions for a fixed task
passes the benchmark's tests.**

Construct

Operationalization

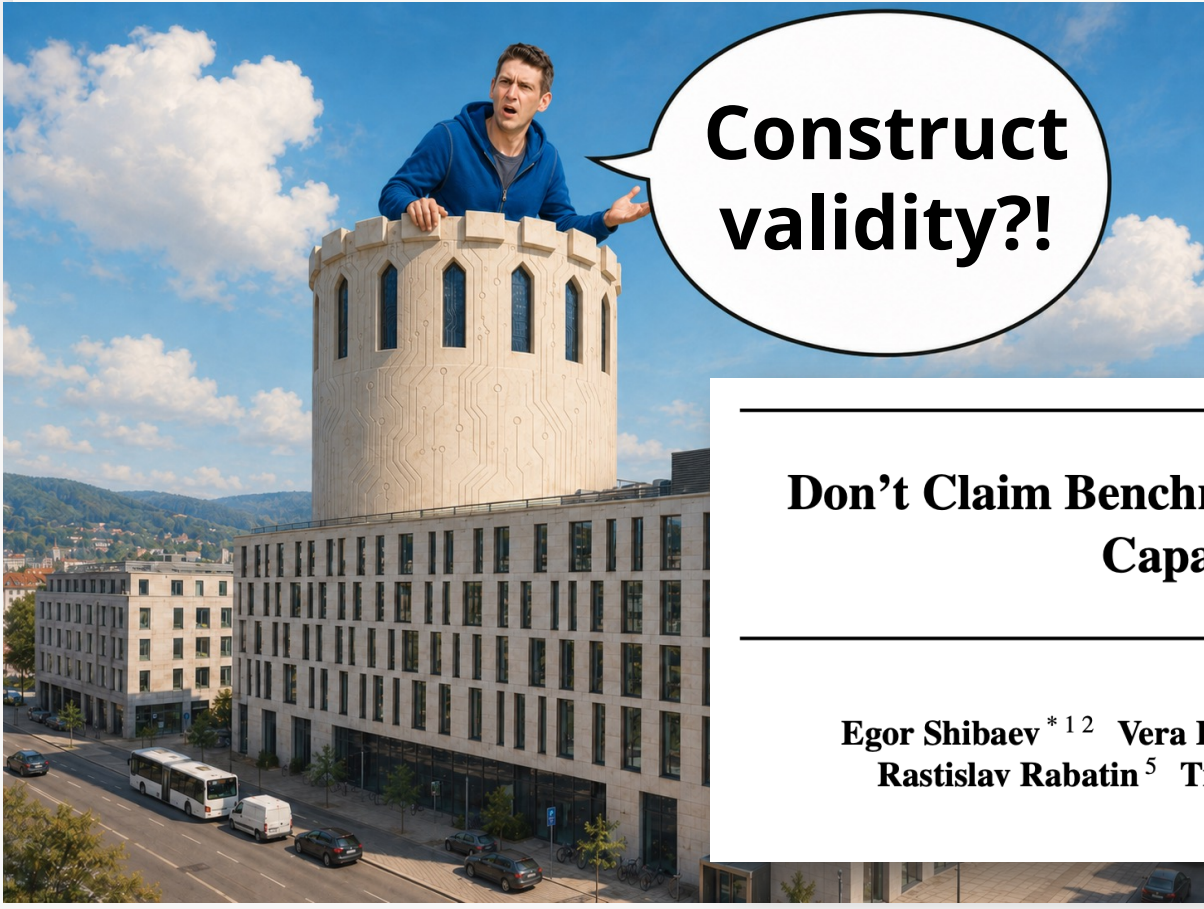
- Just Python.
- Just pure functions.
- No modules.
- No existing codebase.
- No debugging.
- No non-functional properties.



Benchmarks: HumanEval

Frontier intelligence at 2x the speed

Claude 3.5 Sonnet sets new industry benchmarks for graduate-level reasoning



Construct validity?!

coding proficiency

g nuance, humor, and
-quality content with a

- Just Python.
- Just pure functions.
- No modules.
- No existing codebase.
- No debugging.
- No non-functional properties.

pass@k estimates the probability that at least one of k generated

Don't Claim Benchmark-Oriented Optimization Improves General Coding Capability — Diverse Evaluation Is Required

Egor Shibaev^{*12} Vera Kudrevskaia^{*12} Timur Galimzyanov^{*1} Mikhail Evtikhiev^{*3} Ana Terna⁴
Rastislav Rabatin⁵ Timur Kudashev¹² Timofey Bryksin³ Arina Puchkova¹ Patrik Bartak⁴
Egor Bogomolov⁴ Sergey Titov⁴

<https://openreview.net/pdf?id=dQs1srLmGN>

Benchmarks: HumanEval

- HumanEval contained several **incorrect “canonical” solutions**.
- Only **10 assertions per problem on average**, not enough to accurately assess the correctness of solutions.

Is Your Code Generated by ChatGPT Really Correct? **Rigorous Evaluation of Large Language Models for Code Generation**

Jiawei Liu^{1*} Chunqiu Steven Xia^{1*} Yuyao Wang² Lingming Zhang¹

University of Illinois Urbana-Champaign¹ Nanjing University²

{jiawei6, chunqiu2, lingming}@illinois.edu yuyao6@outlook.com

<https://arxiv.org/abs/2305.01210>

“[...] continued pre-training on a dataset containing even a single epoch of the test split for HumanEval **boosted scores from 74.4% to 89.0%, highlighting the danger of data contamination.**”

Google DeepMind

Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context

Gemini Team, Google¹

<https://arxiv.org/pdf/2403.05530>

Benchmarks: HumanEval

- HumanEval contained several **incorrect “canonical” solutions**.

- Only **10** assessments on average, not enough to assess the code quality.

“[...] continued pre-training on a dataset containing even a single

test split for posted scores from, highlighting the contamination.”

Separating signal from noise in coding evaluations

Through a detailed audit, we find widespread task issues in SWE-Bench Pro and estimate that ~30% of the tasks are broken.

Is Your Code Generated by AI?
Rigorous Evaluation for

Jiawei Liu^{1*} Chunqiu Steven Xia^{1*} Yuyao Wang² Lingming Zhang¹

¹University of Illinois Urbana-Champaign ²Nanjing University

{jiawei6, chunqiu2, lingming}@illinois.edu yuyao6@outlook.com

context

Gemini Team, Google¹

<https://arxiv.org/pdf/2403.05530>

<https://arxiv.org/abs/2305.01210>

Newer Benchmarks, Same Problems

Open Benchmarks for Closed Models



<https://knowyourmeme.com/memes/obama-awards-obama-a-medal>

(1) Software Engineering (SE) $\gg$ Programming



(2) AI “solves” Programming $\nRightarrow$ AI “solves” SE



(3) Agentic AI is a Software Systems Innovation

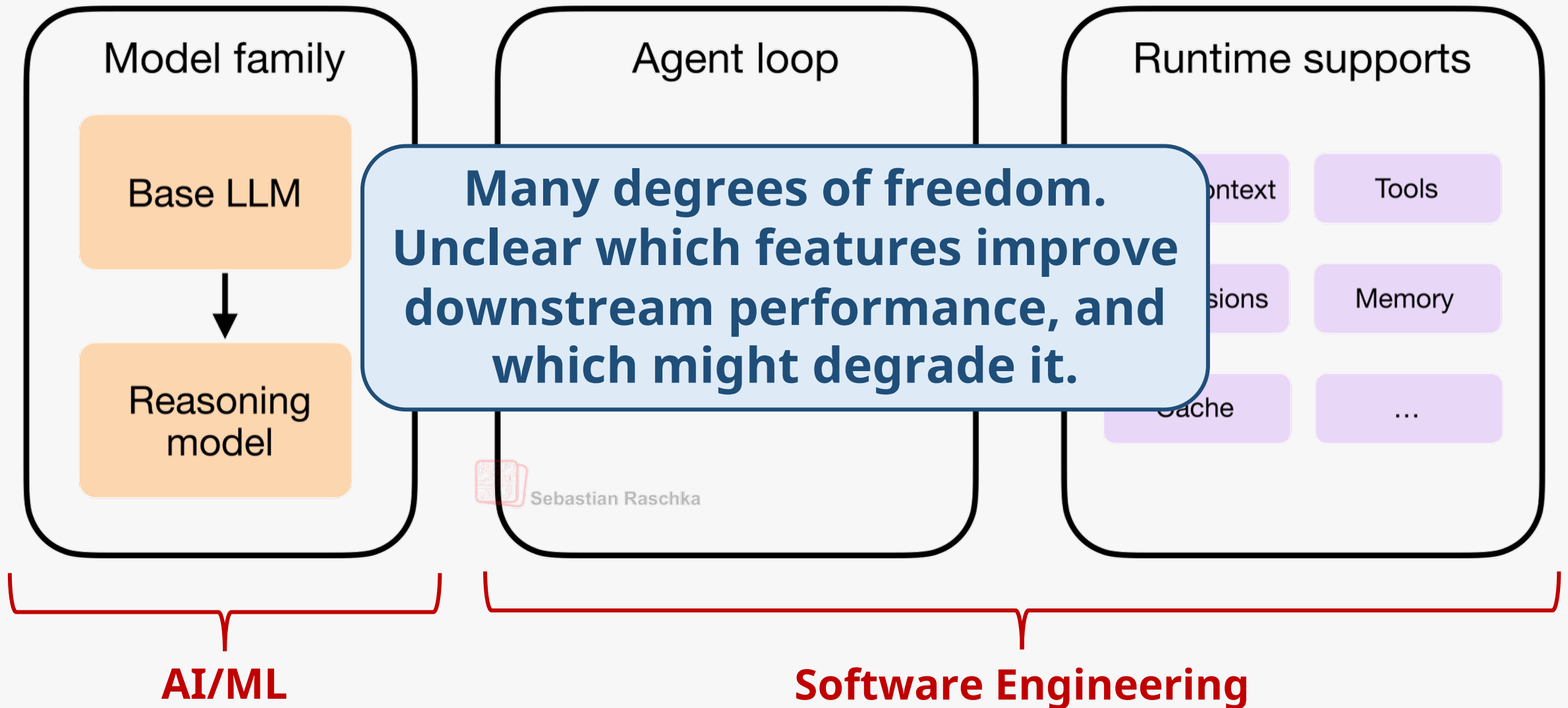


(4) We Need More Solid Empirical Evidence



Problem: We don't even know the “simple” things yet...

Agentic Coding Assistants



<https://magazine.sebastianraschka.com/p/components-of-a-coding-agent>

Configuring Agentic AI Coding Tools: An Exploratory Study

Matthias Galster University of Bamberg Bamberg, Germany mgalster@ieee.org	Syedmoein Mohsenimofidi Heidelberg University Heidelberg, Germany s.mohsenimofidi@uni-heidelberg.de	Jai Lal Lulla Singapore Management University Singapore, Singapore jailal.l.2025@phdcs.smu.edu.sg
Muhammad Auwal Abubakar University of Bamberg Bamberg, Germany muhammad.abubakar@uni-bamberg.de	Christoph Treude Singapore Management University Singapore, Singapore ctreude@smu.edu.sg	Sebastian Baltes Heidelberg University Heidelberg, Germany sebastian.baltes@uni-heidelberg.de

<https://assets.empirical-software.engineering/pdf/aiware26-agents-configuration.pdf>

User Misconceptions of LLM-Based Conversational Programming Assistants

Gabrielle O'Brien University of Michigan Ann Arbor, Michigan, USA elleobri@umich.edu	Antonio Pedro Santos Alves Pontifical Catholic University of Rio de Janeiro Rio de Janeiro, Brazil apsalves@inf.puc-rio.br	Sebastian Baltes Heidelberg University Heidelberg, Germany sebastian.baltes@uni-heidelberg.de
Grischa Liebel Reykjavik University Reykjavik, Iceland grischal@ru.is	Mircea Lungu IT University of Copenhagen Copenhagen, Denmark mlun@itu.dk	Marcos Kalinowski Pontifical Catholic University of Rio de Janeiro Rio de Janeiro, Brazil kalinowski@inf.puc-rio.br

<https://assets.empirical-software.engineering/pdf/jaws26-llm-misconceptions.pdf>

Can We Classify Flaky Tests Using Only Test Code? An LLM-Based Empirical Study

Alexander Berndt  Heidelberg University alexander.berndt@uni-heidelberg.de	Vekil Bekmyradov  SAP vekil.bekmyradov@sap.com	Rainer Gemulla  University of Mannheim rgemulla@uni-mannheim.de
Marcus Kessel  University of Mannheim	Thomas Bach  SAP	Sebastian Baltes  Heidelberg University

On the Impact of AGENTS.md Files on the Efficiency of AI Coding Agents

Jai Lal Lulla Singapore Management University Singapore, Singapore jailal.l.2025@phdcs.smu.edu.sg	Syedmoein Mohsenimofidi Heidelberg University Heidelberg, Germany s.mohsenimofidi@uni-heidelberg.de	Matthias Galster University of Bamberg Bamberg, Germany mgalster@ieee.org
Jie M. Zhang King's College London London, United Kingdom jie.zhang@kcl.ac.uk	Sebastian Baltes Heidelberg University Heidelberg, Germany sebastian.baltes@uni-heidelberg.de	Christoph Treude Singapore Management University Singapore, Singapore ctreude@smu.edu.sg

<https://assets.empirical-software.engineering/pdf/jaws26-agents.md-efficiency.pdf>

The Impact of Configuring Agentic AI Coding Tools on Build-vs-Buy Decisions: A Study Protocol

Jai Lal Lulla  
Singapore Management University, Singapore

Matthias Galster  
University of Bamberg, Germany

Jie M. Zhang  
King's College London, United Kingdom

Sebastian Baltes  
Heidelberg University, Germany

Christoph Treude  

On the Need to Rethink Trust in AI Assistants for Software Development: A Critical Review

Sebastian Baltes^{†*}, Timo Speith^{†**}, Brenda Chiteri[†], Syedmoein Mohsenimofidi[†],
Shalini Chakraborty[‡], Daniel Buschek[‡]

[†]Heidelberg University, Germany

[‡]University of Bayreuth, Germany

[†]{sebastian.baltes, brenda.chiteri, s.mohsenimofidi}@uni-heidelberg.de

[‡]{timo.speith, s.chakraborty, daniel.buschek}@uni-bayreuth.de

<https://assets.empirical-software.engineering/pdf/tse26-trust-ai.pdf>

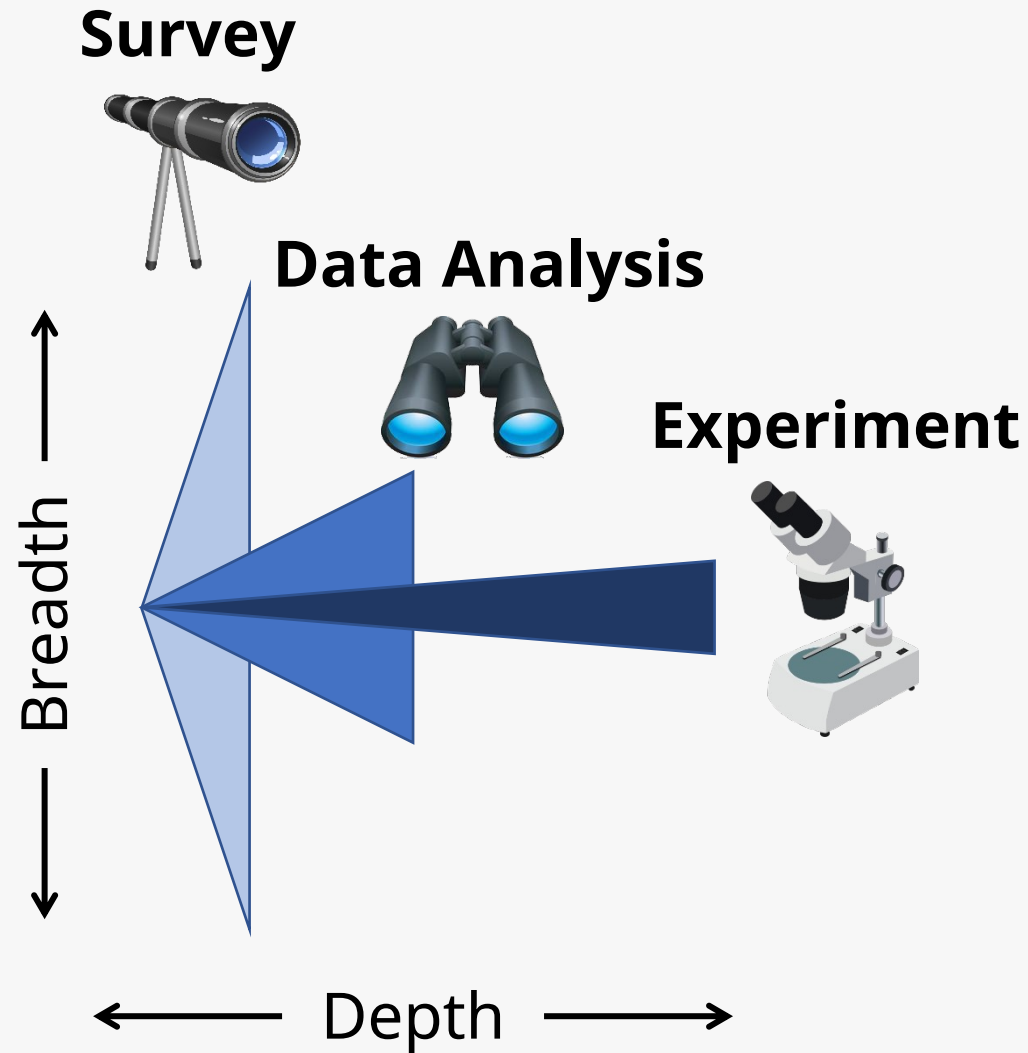
Selected Empirical Studies

Research Methods: Overview



- **Online Surveys:** Capture developers' experiences (e.g., [perceived productivity](#)) or their desires and concerns (e.g., regarding [AI-supported development](#)) using self-reports.
- **Usage Data Analyses:** Analyze real-world usage of tools (e.g., [GitHub Copilot](#)).
- **Controlled Experiments:** Try to assess the impact of "treatments" such as the availability of GenAI tools, usually on relatively small isolated tasks (e.g., in [consulting](#) or [software development](#)).
- **Quasi-experimental Time Series Analyses:** Based on available data, divide time into before/after AI usage and analyze differences (on the level of [projects](#), teams, [individual developers](#)).

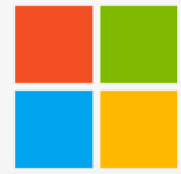
Research Methods: (Incomplete) Overview



Research Methods: There's more...

- **Theory Building** based on interviews (e.g., on [how developers interact with code-generating tools](#)).
- **Comparative Studies** that are not necessarily controlled experiments with random assignment (e.g., [comparing human and AI-generated solutions of algorithmic tasks](#)).
- ...

Desires/Concerns: Survey at Microsoft



Toward Effective AI Support for Developers

A SURVEY OF DESIRES AND CONCERNS

<https://dl.acm.org/doi/10.1145/3675416>

- **Publication:**
July 2024 (peer-reviewed)
- **Data collection:**
April 2023
- **Participants:**
791 software developers
- **Response time:** ~10 min
- **Incentive:** Win a gift card.

Table 1: Survey items

How long have you worked in your current role?

Single choice

Options

- ☐ Less than 6 months
- ☐ 6 months – 1 year
- ☐ 1-3 years
- ☐ 3-7 years
- ☐ More than 7 years

Which of the following aspects of your job would you most like to see AI tools help with?

Selection list. Must select at least 1 item and at most 3 items). Randomized order.

Options

- ☐ Authoring new code and/or refactoring existing code
- ☐ Tracking and managing your work items
- ☐ Prioritizing your tasks
- ☐ Managing your calendar
- ☐ Assistance getting into a state of flow
- ☐ Helping you relax and/or reduce stress
- ☐ Generating unit, integration and functional tests
- ☐ Debugging your code
- ☐ Performing root cause analysis for bugs and incidents
- ☐ Identifying and fixing compliance requirements
- ☐ Helping you identify and consume learning/training material
- ☐ Writing documentation
- ☐ Parsing your email for action items and/or responding to emails on your behalf
- ☐ Analyzing your code for defects, vulnerabilities or optimizations

continues

- ☐ Evaluating your performance to help you identify growth opportunities
- ☐ Policy Management
- ☐ Managing/Acquiring permissions
- ☐ Clarifying requirements
- ☐ Other

Please describe how you envision AI helping with what you selected in the previous questions?

Open text. Optional.

What worries you the most about integrating AI into your daily workflows?

Single choice. Randomized order

Options

- ☐ Automating away your job
- ☐ Increasing bias in the workplace
- ☐ Introducing defects or vulnerabilities into your work
- ☐ Monetary costs to deploy within your team/organization
- ☐ Impact on climate change
- ☐ Being more gimmicky than helpful
- ☐ Having to change the workflows you have established and are used to
- ☐ Diminishing connections with your peers
- ☐ Causing your skills to atrophy The time it will take to learn how to use new AI tools
- ☐ Automation causing compensation to decrease
- ☐ Other

Is there anything else you would like to share about your thoughts on the future of AI in software engineering?

Open text. Optional

Desires/Concerns: Survey at Microsoft

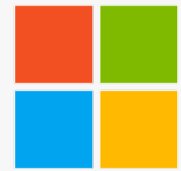


FIGURE 1: RESPONSES TO QUESTION ABOUT DESIRE FOR AI HELP

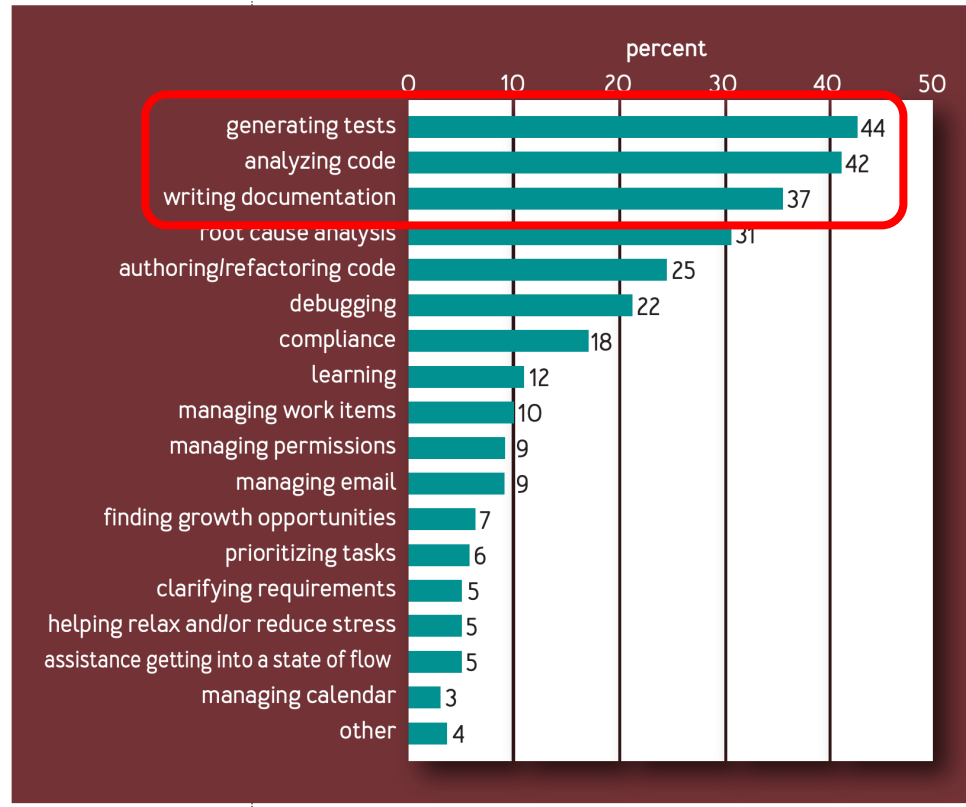
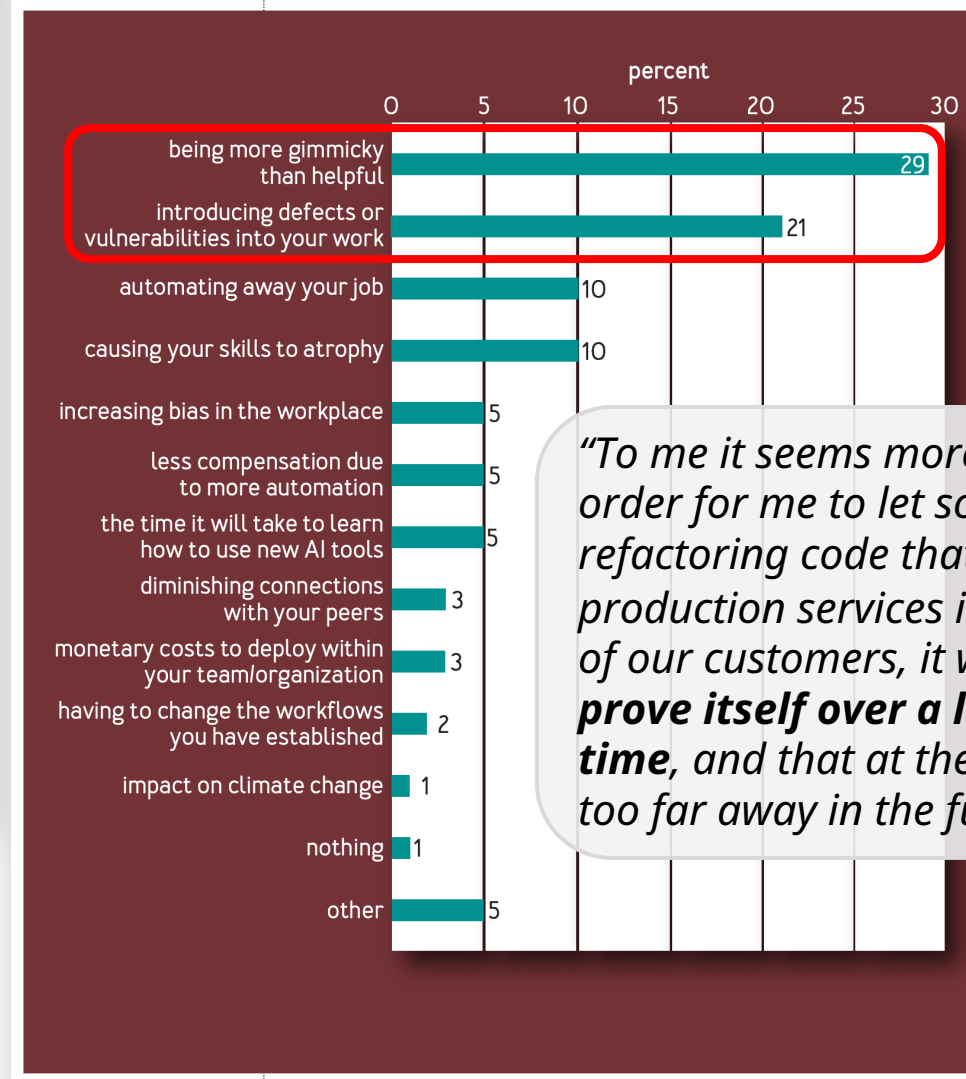


FIGURE 2: RESPONSES TO QUESTION ABOUT DESIRE FOR AI CONCERNS



*"To me it seems more like a **gimmick**. In order for me to let something [...] refactoring code that is used in production services impacting millions of our customers, it would need to **prove itself over a long period of time**, and that at the very least seems too far away in the future."*



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



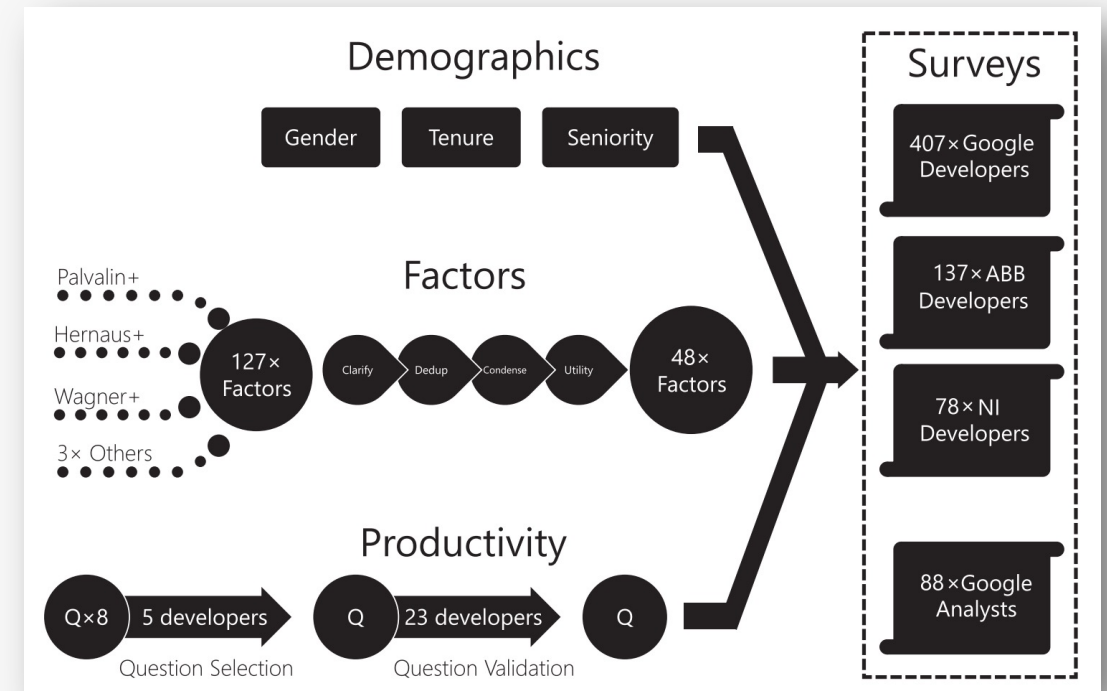
What Predicts Software Developers' Productivity?

Emerson Murphy-Hill ^{ID}, Ciera Jaspan ^{ID}, Caitlin Sadowski, David Shepherd ^{ID}, Michael Phillips ^{ID}, 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



Self-reports: Correlation Study Across 3 Companies



We asked respondents about their productivity by rating their agreement with the following statement:

I regularly reach a high level of productivity.

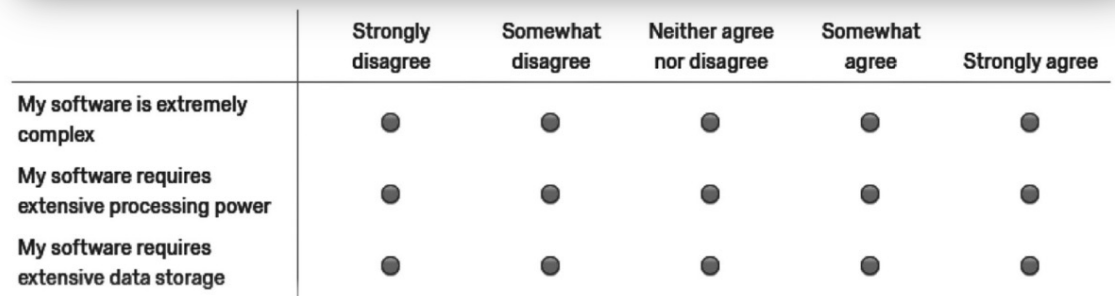


Fig. 3. Example factor questions from the survey.

	Google (n=407)		ABB (n=137)		NI (n=78)		estimate μ (σ)	Google Analysts (n=88)		
	estimate	error	estimate	error	estimate	error		estimate	error	diff
I am enthusiastic about my job	0.414 *	0.049	0.386 *	0.090	0.484 *	0.089	0.43 (0.051)	0.422	0.097	-0.01
People on my project are supportive of new ideas	0.337 *	0.059	0.318 *	0.090	0.312 *	0.102	0.32 (0.013)	0.461	0.133	-0.12
My job allows me to make decisions about what methods I use to complete my work	0.298 *	0.058	0.360 *	0.093	0.237	0.106	0.30 (0.061)	0.365	0.157	-0.07
My job allows me to make my own decisions about managing my time	0.293 *	0.056	0.318 *	0.078	0.237	0.121	0.28 (0.042)	0.487	0.106	-0.19
People who manage my project are highly capable, efficient, thorough, communicative, and cooperative	0.318 *	0.053	0.241 *	0.084	0.264	0.112	0.27 (0.04)	0.287	0.105	0.03
The information supplied to me (bug reports, user stories, etc.) is accurate	0.233 *	0.061	0.161	0.087	0.418 *	0.111	0.27 (0.132)	-	-	-
I feel positively about other people on my project	0.291 *	0.063	0.240	0.103	0.278	0.148	0.27 (0.027)	0.611	0.135	-0.32
My job allows me to use my personal judgment in carrying out my work	0.372 *	0.058	0.242 *	0.090	0.172	0.100	0.26 (0.101)	0.519	0.127	-0.15
My project resolves conflicts quickly	0.295 *	0.048	0.207	0.085	0.272	0.115	0.26 (0.046)	0.456	0.111	-0.16
People who write code for my software are highly capable, efficient, thorough, communicative, and cooperative	0.348 *	0.058	0.177	0.084	0.245	0.124	0.26 (0.086)	-	-	-
I receive useful feedback about my job performance	0.245 *	0.050	0.262 *	0.076	0.259 *	0.091	0.26 (0.009)	0.220	0.125	0.02
My job requires me to use a number of complex or high-level skills	0.304 *	0.055	0.246 *	0.095	0.193	0.136	0.25 (0.056)	0.286	0.130	0.02
My job involves a great deal of task variety	0.163 *	0.057	0.235 *	0.089	0.336	0.133	0.24 (0.087)	-0.010	0.128	0.17
People who work on my software's requirements and design are highly capable, efficient, thorough, communicative, and cooperative	0.289 *	0.050	0.174	0.076	0.267 *	0.094	0.24 (0.061)	-	-	-
I use the best tools and practices to develop my software	0.445 *	0.052	0.190	0.082	0.095	0.109	0.24 (0.181)	0.501	0.144	-0.06
Knowledge flows adequately between the key persons in our project	0.251 *	0.048	0.222 *	0.080	0.198	0.106	0.22 (0.026)	0.256	0.106	-0.01
I prepare for meetings	0.222 *	0.050	0.121	0.095	0.292	0.102	0.21 (0.086)	0.270	0.156	-0.05
My project's meeting practices are efficient	0.233 *	0.048	0.162	0.082	0.238	0.097	0.21 (0.042)	0.297	0.097	-0.06
I have few interruptions or distractions while working	0.242 *	0.043	0.160	0.068	0.202	0.100	0.20 (0.041)	0.194	0.090	0.05
My project's bug finding process is efficient and effective	0.294 *	0.047	0.092	0.076	0.217	0.100	0.20 (0.102)	-	-	-
The software process my project uses is well-defined	0.309 *	0.046	0.121	0.072	0.165	0.114	0.20 (0.098)	0.321	0.106	-0.01
I use the best tools and practices to develop my software	0.252 *	0.062	0.174	0.090	0.155	0.119	0.19 (0.051)	0.413	0.163	-0.16
My software's platform (software stack and hardware stack) changes rapidly	0.235 *	0.036	0.199 *	0.061	0.139	0.081	0.19 (0.048)	0.178	0.083	0.06
My software's requirements change frequently	0.214 *	0.047	0.067	0.078	0.292	0.107	0.19 (0.114)	0.146	0.109	0.07
I have extensive experience with my software's platform (software stack and hardware stack)	0.310 *	0.052	0.030	0.074	0.221	0.144	0.19 (0.143)	-	-	-
Security vulnerabilities, changes in requirements, etc.)	0.201 *	0.047	0.130	0.078	0.216	0.113	0.18 (0.046)	-	-	-
Security vulnerabilities, changes in requirements, etc.)	0.313 *	0.050	0.062	0.087	0.141	0.104	0.17 (0.128)	-	-	-
Skills and programming languages used in my software	0.161 *	0.053	0.144	0.083	0.174	0.123	0.16 (0.015)	-	-	-
People in the company besides the people on my project	0.121 *	0.042	0.093	0.073	0.263 *	0.083	0.16 (0.091)	0.153	0.095	-0.03
I can work effectively away from my desk	0.156 *	0.040	0.128	0.071	0.073	0.092	0.12 (0.042)	0.005	0.104	0.15
Using other software similar to the one I'm working on	0.173 *	0.044	0.037	0.079	0.107	0.105	0.11 (0.068)	0.124	0.114	0.05
Context switching is a necessary part of my job	0.077	0.058	-0.027	0.081	0.217	0.113	0.09 (0.123)	-0.029	0.118	0.11
People on my project are physically collocated	0.100 *	0.040	0.015	0.063	0.087	0.086	0.07 (0.046)	0.150	0.085	-0.05
My project deadlines are tight	0.061	0.045	0.024	0.076	0.097	0.125	0.06 (0.037)	0.011	0.120	0.05
Direct access to specific hardware to test my software	0.079 *	0.033	0.041	0.058	-0.031	0.082	0.03 (0.056)	-	-	-
Used widely and heavily by other software developers	0.117 *	0.038	-0.053	0.066	0.008	0.089	0.02 (0.086)	-	-	-
My software's requirements change frequently	-0.033	0.050	0.010	0.079	0.076	0.115	0.02 (0.055)	-	-	-
and maintain the data necessary to test my software	0.038	0.040	0.025	0.076	-0.009	0.097	0.02 (0.024)	-	-	-
My software requires extensive processing power	0.027	0.041	0.044	0.071	-0.040	0.110	0.01 (0.045)	-	-	-
Doing out tasks that require uninterrupted concentration	0.006	0.039	0.002	0.061	-0.008	0.092	0.00 (0.007)	-0.110	0.088	0.12
My project has many people working on it	0.002	0.043	0.035	0.062	-0.039	0.100	0.00 (0.037)	0.037	0.107	-0.04
Requirements are high (e.g., privacy, legal, environmental, etc)	-0.044	0.043	0.058	0.076	-0.018	0.119	0.00 (0.053)	-	-	-
My software requires extensive data storage	-0.018	0.039	0.008	0.070	-0.015	0.105	-0.01 (0.014)	-	-	-
My software is extremely complex	-0.013	0.053	0.115	0.084	-0.143	0.111	-0.01 (0.129)	-	-	-
My software's platform (e.g. development environment, software stack, hardware stack) changes rapidly	-0.005	0.040	-0.058	0.068	-0.035	0.086	-0.03 (0.027)	0.014	0.091	-0.02
Extensive documentation is required to use my software at different points in its lifecycle	0.054	0.046	-0.058	0.083	-0.166	0.095	-0.06 (0.11)	-	-	-
Personnel turnover on my project is high	-0.043	0.047	-0.069	0.076	-0.085	0.111	-0.07 (0.022)	-	-	-
	-0.040	0.045	-0.153	0.079	-0.160	0.102	-0.12 (0.068)	-0.096	0.097	0.06

Fig. 4. 48 factors' correlation with developers' and analysts' self-rated productivity at three companies.

Comparative Study: Uplevel Report

Can Generative AI Improve Developer Productivity?

HERE'S WHAT THE REAL-LIFE DATA SUGGESTS



- **Publication:**
September 2024 (report)
 - Will AI help developers ship faster? Can it help them write better code and avoid burnout?
 - Not yet for this population. But innovation moves fast, and GitHub reports that Copilot does improve developer satisfaction. Engineering leaders may benefit from adopting a
- **Data Collection:**
Jan-Apr 2023 vs. Jan-April 2024
- **Participants:**
800 developers
- **Goal:**
Comparing development metrics before/after introduction of Copilot <https://uplevelteam.com/blog/ai-tools-in-software-development>
- **Metrics:**
 - **PR cycle time:** no significant change
 - **Bug rate:*** 41% increase with Copilot
 - **Working outside standard hours:**
Decreased for both groups

* unclear how measured

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

- **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 β 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>

-2000 Lines of Code

In early 1982, the Lisa software team was trying to buckle down for the big push to ship the software within the next six months. Some of the managers decided that it would be a good idea to track the progress of each individual engineer in terms of the amount of code that they wrote from week to week. They devised a form that each engineer was required to submit every Friday, which included a field for the number of lines of code that were written that week.

Bill Atkinson, the author of Quickdraw and the main user interface designer, who was by far the most important Lisa implementer, thought that lines of code was a silly measure of software productivity. He thought his goal was to write as small and fast a program as possible, and that the lines of code metric only encouraged writing sloppy, bloated, broken code.

He recently was working on optimizing Quickdraw's region calculation machinery, and had completely rewritten the region engine using a simpler, more general algorithm which, after some tweaking, made region operations almost six times faster. As a by-product, the rewrite also saved around 2,000 lines of code.

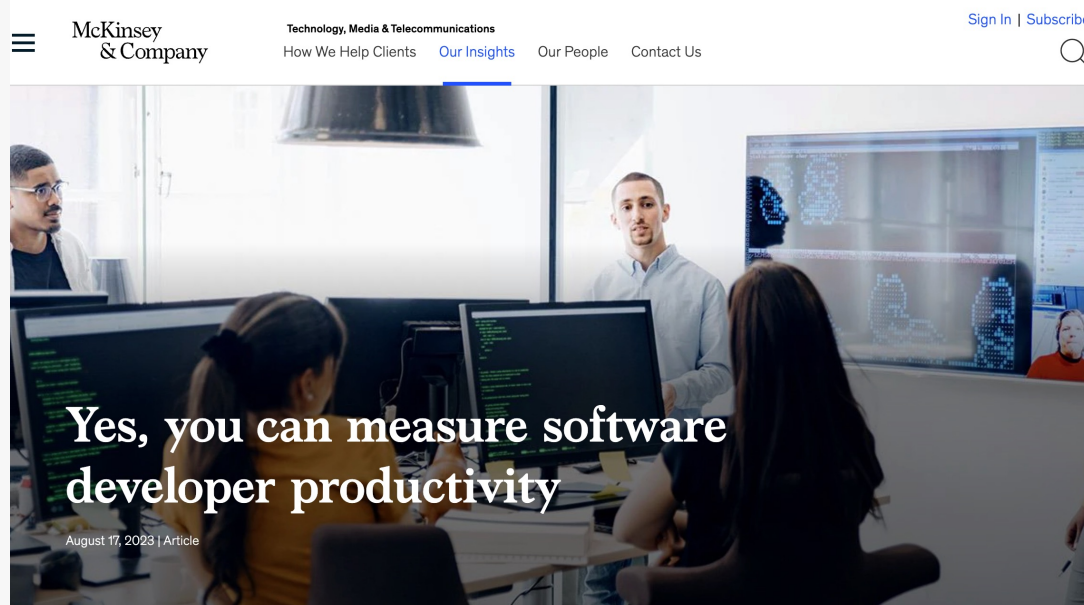
He was just putting the finishing touches on the optimization when it was time to fill out the management form for the first time. When he got to the lines of code part, he thought about it for a second, and then wrote in the number: -2000.

I'm not sure how the managers reacted to that, but I do know that after a couple more weeks, they stopped asking Bill to fill out the form, and he gladly complied.

Summary

- **AI assistants do affect software development activity.**
(exact impact on productivity is unclear)
- **Too many studies focus on self-assessed productivity alone.**
(known to mainly reflect non-technical aspects)
- **Too few longitudinal studies.**
- **Too few studies with a more holistic view on software development.**

Back to the Consulting Industry...



Controlled Experiments: Boston Consulting Group

Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality

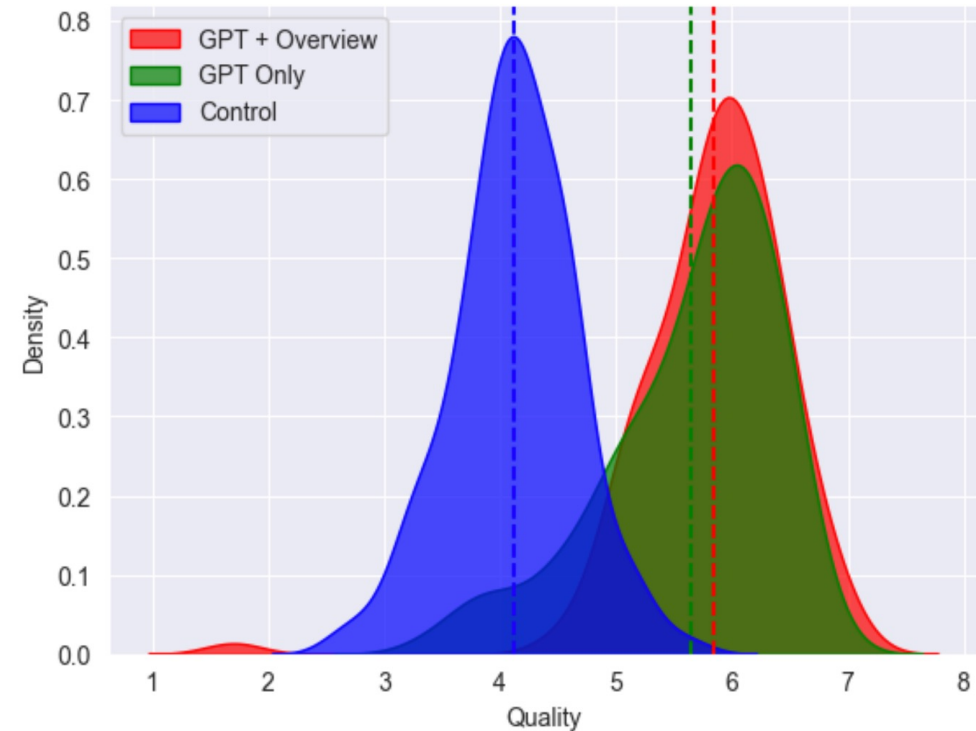
- **Publication:**
September 2023
(preprint)
- **Data collection:**
April 2023
- **Participants:**
758 BCG consultants
- **Experiment duration:**
5 hours
- **Treatments:**
GPT-4 and GPT-4 plus prompt engineering introduction.
- **Performance:**
Assessed by two human graders.
- **Incentive:**
Best participants received gift in recognition as part of their performance review.

<https://www.hbs.edu/ris/download.aspx?name=24-013.pdf>

Controlled Experiments: Boston Consulting Group

“For each one of a set of **18 realistic consulting tasks** within the frontier of AI capabilities, consultants using AI were significantly more productive (they completed 12.2% **more tasks** on average, and completed tasks 25.1% **more quickly**), and produced significantly **higher quality results** (more than 40% higher quality compared to a control group).”

Figure 2: Performance Distribution - Inside the Frontier



Notes: This figure displays the full distribution of performance in the experimental task inside the frontier for subjects in the three experimental groups (red for subjects in the GPT+Overview condition; green for subjects in the GPT Only condition; blue for subjects in the control condition).

Controlled Experiments: Boston Consulting Group

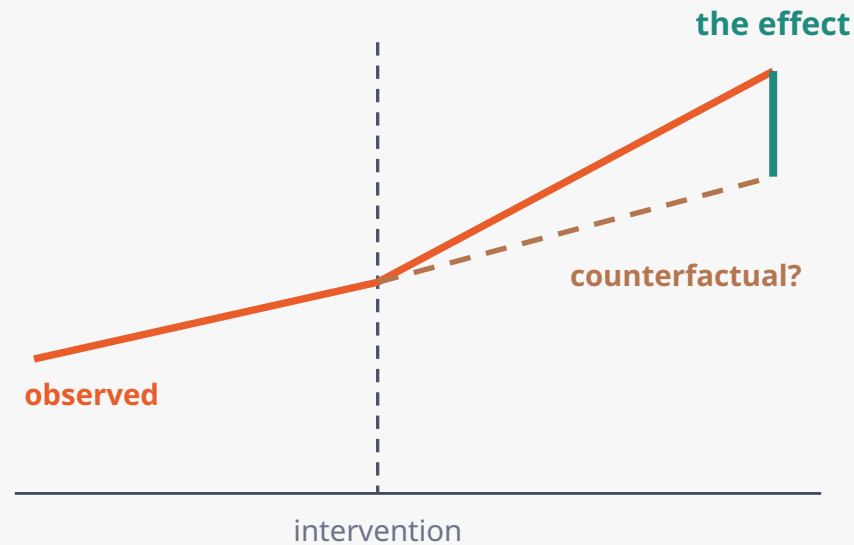
*“You are working for a **footwear company** in the unit **developing new products**. Your boss asked you to **present an idea for a new product** at the next manager’s meetings. Please, respond to the questions below.*

- 1. Generate ideas for a new shoe aimed at a specific market or sport that is underserved. Be creative, and give at least 10 ideas.*
- 2. Pick the best idea, and explain why, so that your boss and other managers can understand your thinking.*
- 3. Describe a potential prototype shoe in vivid detail in one paragraph (3-4 sentences).*
- 4. Come up with a list of steps needed to launch the product. Be concise but comprehensive.*
- 5. Come up with a name for the product: consider at least 4 names, write them down, and explain the one you picked.*

...”

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

Longitudinal Quasi-Experimental Designs



Examples

Scenario: An organization enables an **AI coding assistant** for some teams in July 2025 (→ treatment). Did it change **throughput**?

1 Before vs. after Treated teams only

13.0 vs. 17.5

+4.5

PRs per dev per month

Confounded by time. Q3 also brought a new build pipeline and the end of a release freeze. The trend was already rising independent of the tool rollout.

2 Treated vs. control After the rollout only

12.5 vs. 17.5

+5.0

PRs per dev er month

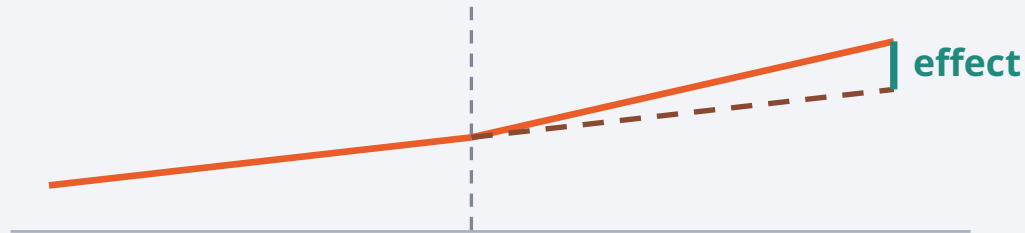
Confounded by team selection. The teams that opted in to use the AI tool were already shipping faster than other teams in June, i.e., before tool adoption.

Problem: What's the proper **counterfactual** to compare with?

Longitudinal Quasi-Experimental Designs

1 Interrupted time series

“time-series experiment”, usually analyzed by segmented regression



Needs: one group, ~8–12 observations both before/after

Counterfactual: that group's own pre-trend, extended past the rollout

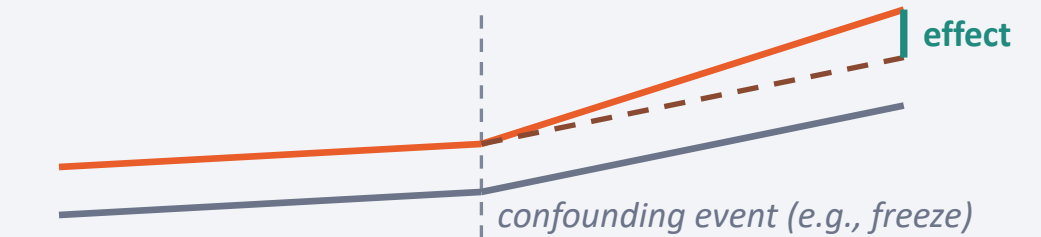
Estimator: fit a line before and after; compare level and slope

Assumes: the pre-trend would have continued

Confounder: anything else that happened between before and after

2 Difference-in-differences

non-equivalent control-group design, usually analyzed by interaction regression



Needs: two groups; one measurement before and one after is enough

Counterfactual: the treated teams, had they moved like the control

Estimator: start at treated rollout level; add the control's gain

Assumes: parallel trends: without rollout, both groups change alike

Confounder: groups that were already diverging

When do both approaches agree?

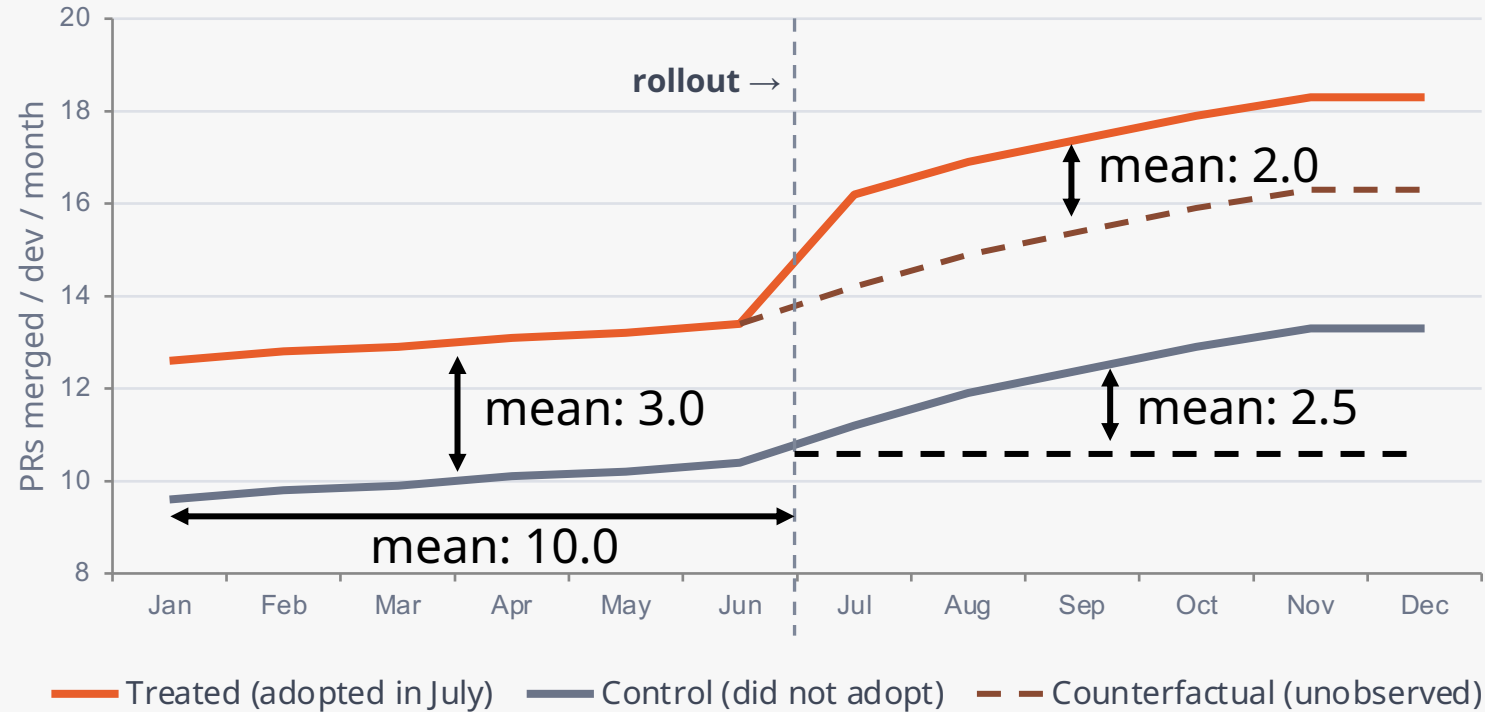
Difference-in-Differences: Example

Mean number of PRs merged per developer per month, averaged over the **six months** either side of the **rollout (2 groups x 2 periods)**.

	Before (Jan-Jun)	After (Jul-Dec)	Difference
Treated teams 24 teams, adopted in July	13.0	17.5	+4.5
Control teams 31 teams, did not adopt	10.0	12.5	+2.5
Difference of differences	4.5 - 2.5 =		+2.0 95% CI 0.5 to 3.5

The teams that adopted gained **2.0 more PRs** per developer per month than they would have, had they tracked the teams that did not adopt.

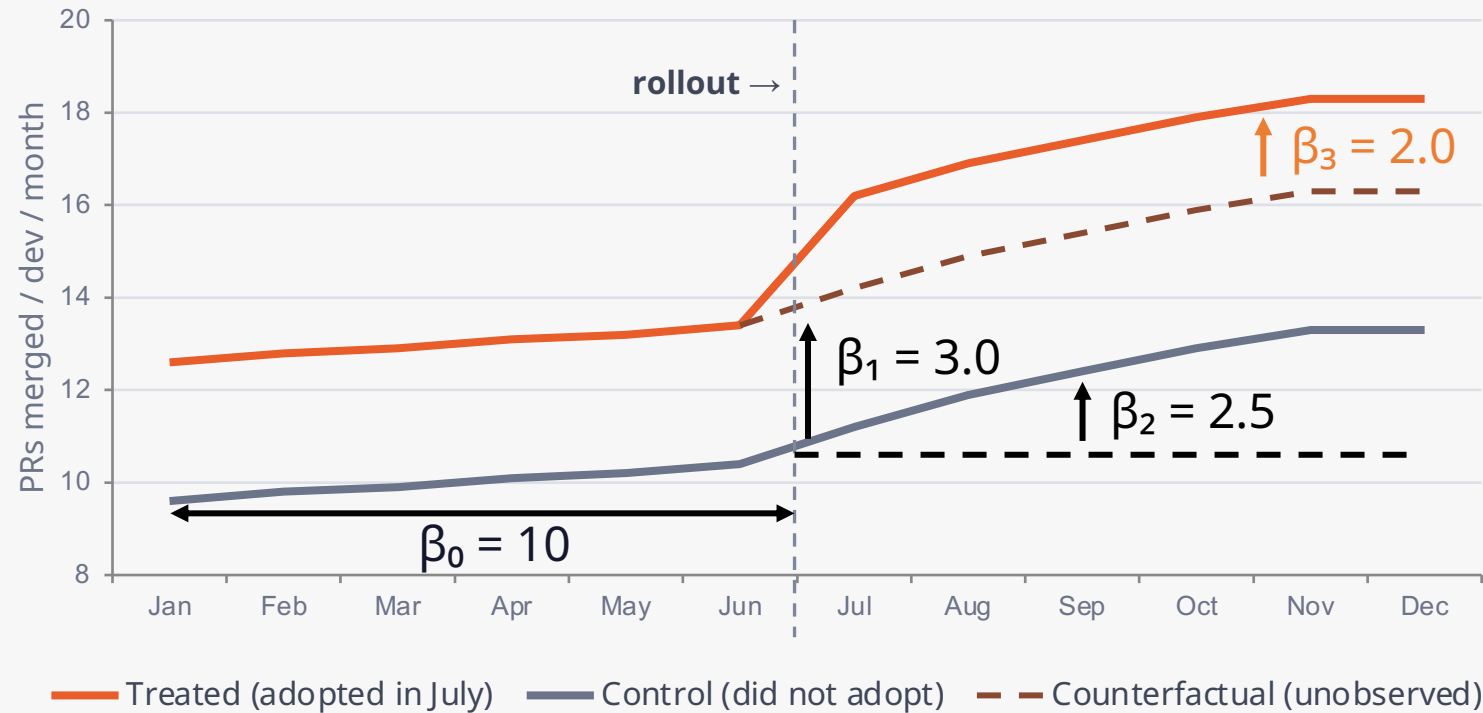
Difference-in-Differences: Example



Difference-in-Differences: Linear Regression

Why do we need this? ?

$$Y_{it} = \beta_0 + \beta_1 \cdot Treated_i + \beta_2 \cdot Post_t + \beta_3 \cdot (Treated_i \cdot Post_t) + \varepsilon_{it}$$

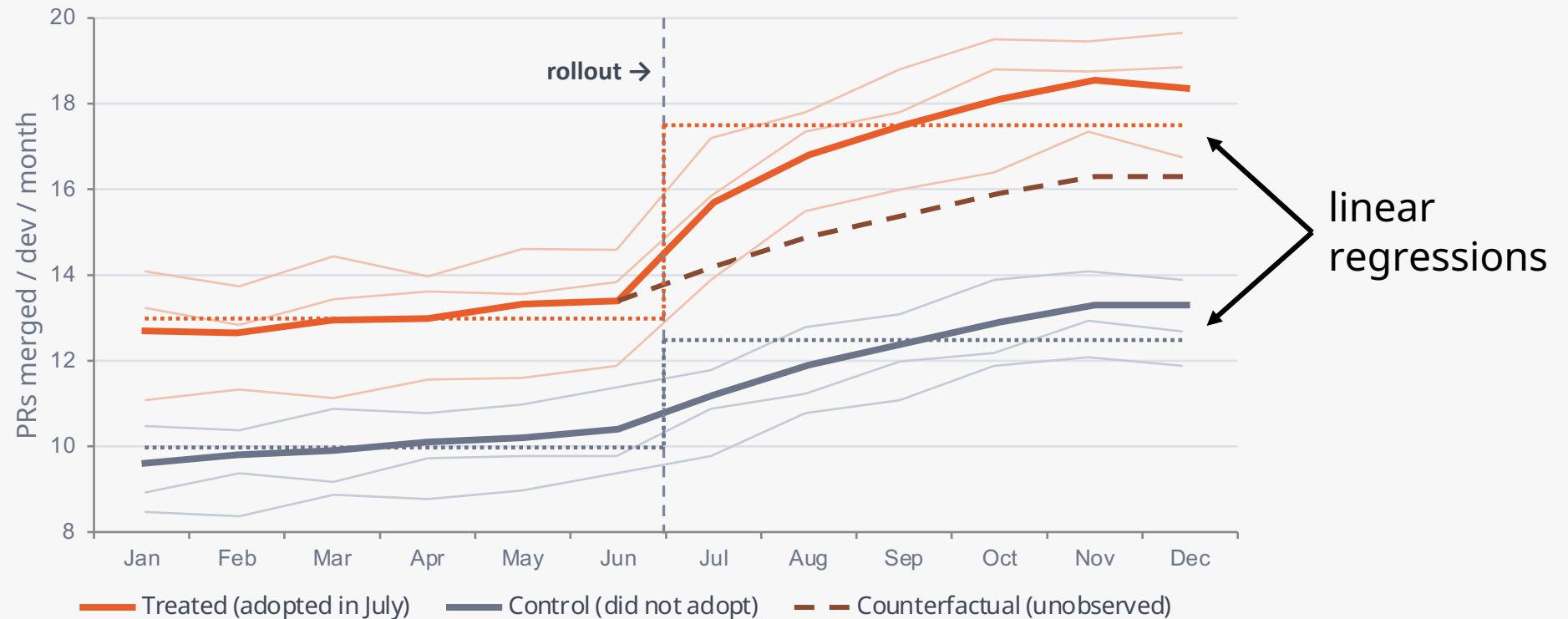


β_3 is the difference in differences

$10.0 + 3.0 + 2.5 + 2.0 = 17.5$
(mean value of treated teams after rollout)

Difference-in-Differences: Linear Regression

$$Y_{it} = \beta_0 + \beta_1 \cdot Treated_i + \beta_2 \cdot Post_t + \beta_3 \cdot (Treated_i \cdot Post_t) + \varepsilon_{it}$$



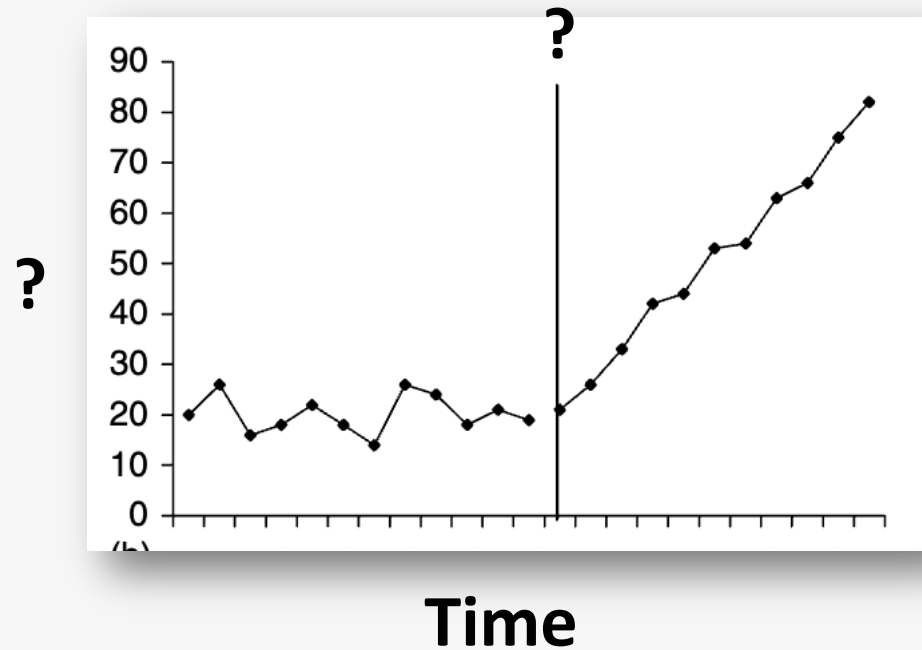
Y_{it} merged PRs/developer/month, for team i in month t .

$Post_t$ **0** for Jan-Jun, **1** for Jul-Dec.

$Treated_i$ **1** if team i treated, **0** otherwise.

$(Treated_i \cdot Post_t)$ **1** only if treated and Jul-Dec.

What to measure over time?



<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8011057/> <https://doi.org/10.1002/0470013192.bsa312>

Difference-in-Differences: Example

<https://cmustrudel.github.io/papers/msr2026he.pdf>

Speed at the Cost of Quality

How Cursor AI Increases Short-Term Velocity and Long-Term Complexity in Open-Source Projects

Hao He

Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Courtney Miller

Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Shyam Agarwal

Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Christian Kästner

Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Bogdan Vasilescu

Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

3.1.2 Identifying GitHub Projects Adopting Cursor. We identify Cursor-adopting repositories and track adoption dates through configuration files in the git history. In the GitHub code search API [14],

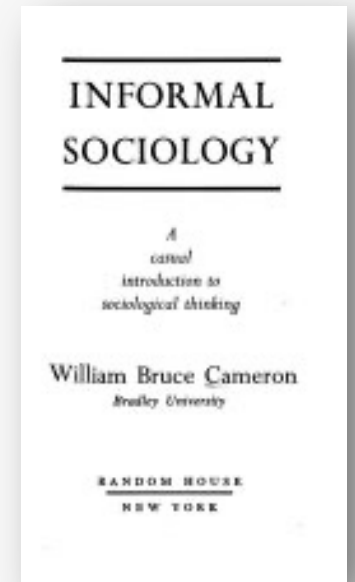
It's actually **more complicated** than this because the authors needed to model a staggered rollout.

- Commits_{it} : Number of commits in repository i at month t ;
- Lines Added_{it} : Total lines added, summed over all commits in repository i at month t .
- $\text{Static Analysis Warnings}_{it}$: Total number of reliability, maintainability, and security issues for repository i at month t , as detected by SonarQube's static analysis. We refer to them as warnings, since static analysis can generate false positives [56]; this metric should be viewed as an estimate of the effort required to review potential issues in a project.
- $\text{Duplicate Line Density}_{it}$: Percentage of duplicated lines in code-base for repository i at month t . SonarQube's definition varies across programming languages, but usually requires at least 10 consecutive duplicate statements or 100 duplicate tokens to mark a block as duplicate [18].

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).



What Can We Measure in a Software Project?

- **Process-related measures:**

- Utilized resources (people, time, money, etc.).
- Number of issues detected during code review.

- **People-related measures:**

- Perceived productivity and wellbeing of team members.
- Time spent on different tasks (self-reports or log data).

- **Product-related measures:**

- Size (LOC, #classes/modules/services, etc.).
- Code Complexity.
- Code Readability.
- Code Quality (coverage, technical debt, etc.).

Measurement?



“Measurement is the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of describing them.”

Kaner and Bond: Software Engineering Metrics ([2004](#))

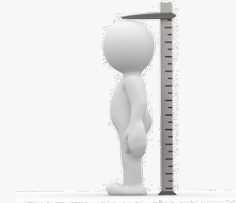
Measurement



- **Measurement:** Mapping from studied object to a scale, e.g.:

height : people $\rightarrow \mathbb{R}^{\geq 0}$

sex : people $\rightarrow \{\text{male, female, intersex}\}$



- **Scale types / levels of measurement (“Skalenniveau”):**
 - **Nominal** (sex, gender).
 - **Ordinal** (minor, ..., major).
 - **Interval** (temp. °C).
 - **Ratio** (temp. K, height).
- **Example from SE:** LOC as a measurement for program size.
- **Metrology:** Science of measurement.

Measure (“Maß”) vs. Metric (“Metrik”)?

- Multiple, sometimes contrary, definitions [exist](#).
- [NIST](#): “We use **measure** for **more concrete** or objective attributes and **metric** for **more abstract**, higher-level, or somewhat subjective attributes.”
- **Measurement:**
 - **Directly observable** quantitative values.
 - Fundamental building blocks of metrics.
- **Metrics:**
 - **Derived** from one or more measurements by interpreting and combining them.
 - Capture more **abstract concepts** such as “quality” or “productivity”.

Measure vs. Metric: Example

- **Target Metric:** Software Quality 
- **Measurements:**
 - Code Coverage [0.0, 1.0]
 - Availability (e.g., of a cloud service) last month [0.0, 1.0]
 - SUS from quarterly user survey [0.0, 1.0]
- **Metric:** $0.2 * \text{Coverage} + 0.4 * \text{Availability} + 0.4 * \text{SUS}$ [0.0, 1.0]
- Unclear to what degree that metric measures the broad **abstract concept** of “software quality.”

Further reading:

<https://arxiv.org/pdf/2603.16012v1>

(slightly different notion of metric/measure than we use)

Software Metrics

Warning



- **Most software metrics are controversial.**
- They are often defined based on plausibility arguments and are rarely rigorously validated.
- **Size of codebase** dominates many metrics.
- **Example:** Evidence indicates that cyclomatic complexity (a popular metric) is merely a [proxy for program size](#).

The GitClear Report



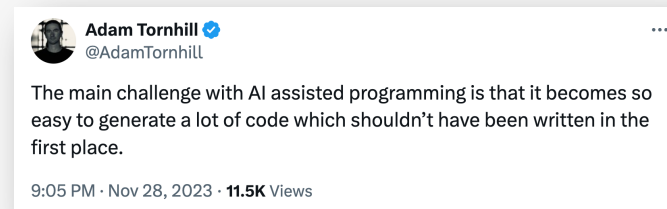
Coding on Copilot

2023 Data Shows Downward Pressure on Code Quality

150m lines of analyzed code + projections for 2024

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

“Code suggestion algorithms are incentivized to propose **suggestions most likely to be accepted**. Code maintainers are incentivized to **minimize the amount of code that needs to be read** (i.e., to understand how to adapt an existing system).”



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

GitClear Report

Burgeoning Churn

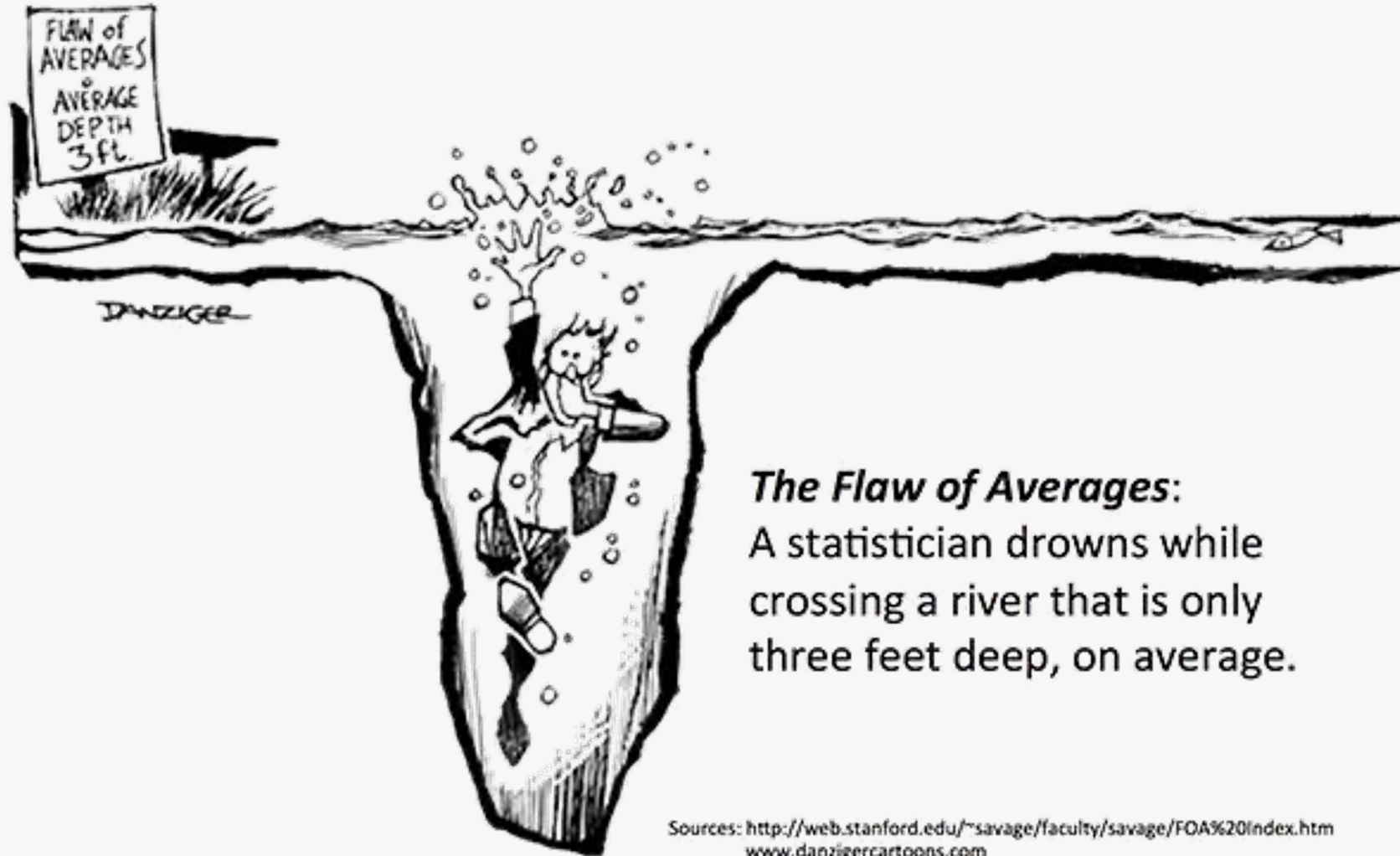
Recall that "Churn" is the percentage of code that was pushed to the repo, then subsequently reverted, removed or updated within 2 weeks. This was a relatively infrequent outcome when developers authored all their own code -- only 3-4% of code was churned prior to 2023, although there is also a hint 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.

More Copy/Pasted Code Implies Future Headaches

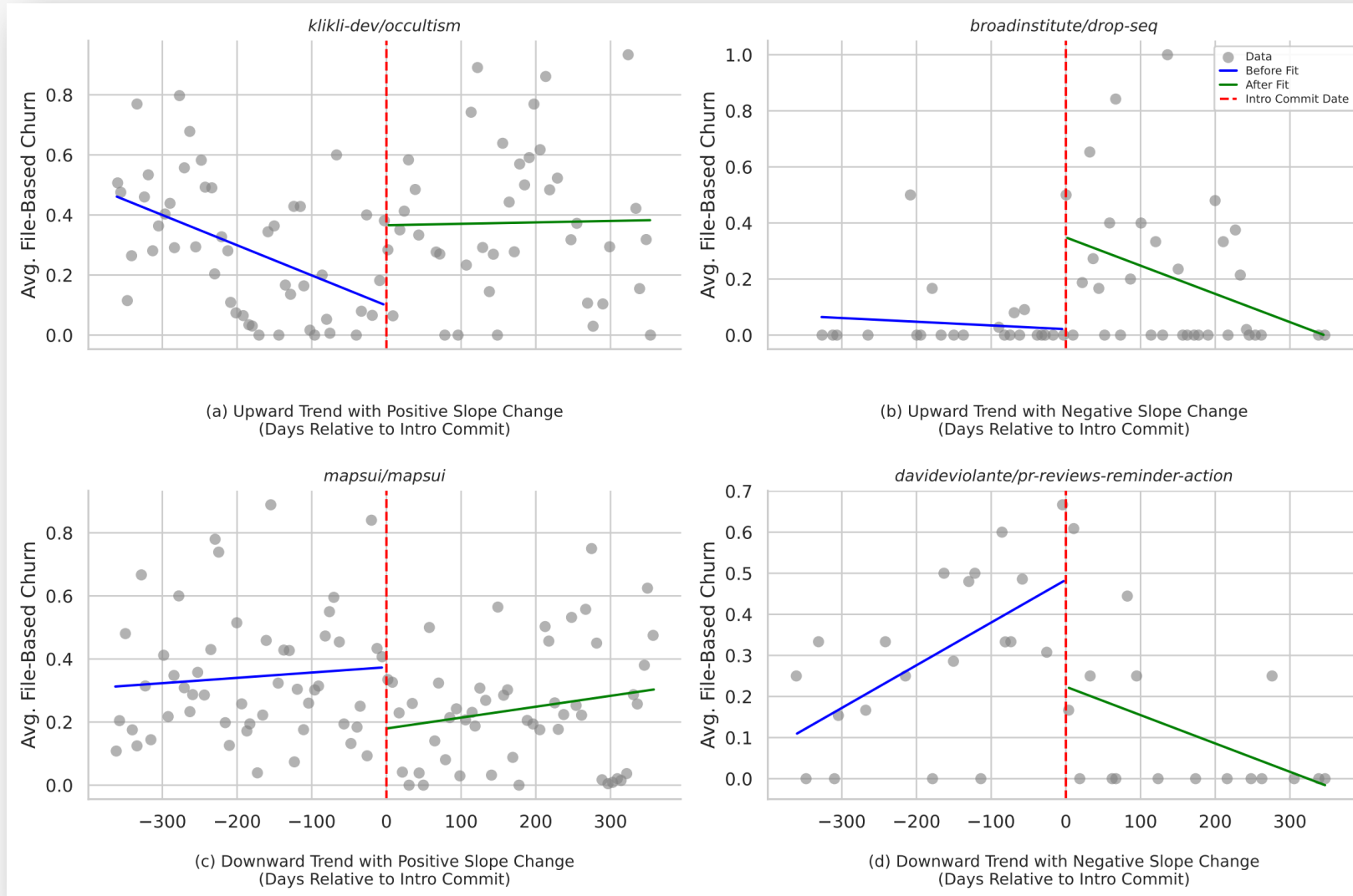
Less Moved Code Implies Less Refactoring, Less Reuse

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

Do not jump to calculating statistics too fast!



We could not replicate this result...



<https://assets.empirical-software.engineering/pdf/tse26-genai-usage.pdf>

Why Are Inaccurate Metrics Problematic?

Management:

“You can't control what you can't measure.”

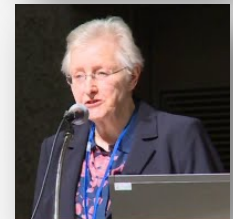
DeMarco, Tom: *Controlling Software Projects*. In *Management Measurement & Estimation* (1982), p. 3.



Common Sense:

“When a measure becomes a target, it ceases to be a good measure.”

Strathern, Marilyn: *Improving ratings: audit in the British University system*. In *European Review*. 5 (3), p. 305–321. (1997); based on Goodhart, Charles: *Problems of Monetary Management: The U.K. Experience*. In Courakis, Anthony S. (ed.). *Inflation, Depression, and Economic Policy in the West*. (1975, published in 1981). Known as “Goodhart’s Law.”



<https://www.youtube.com/watch?v=AcQYFP4PwFs>
<https://www.youtube.com/watch?v=mhNOVuFwIjo>
<https://systemsguild.eu/tom-demarco>

Tom DeMarco's Reflections

Management:

"You can't control what you can't measure."

DeMarco, Tom: *Controlling Software Projects*. In *Management Measurement & Estimation* (1982), p. 3.



“

In my reflective mood, I'm wondering,

- Was its advice correct at the time?
- Is it still relevant? and
- Do I still believe that metrics are a must for any successful software development effort?

My answers are no, no, and no.

--Tom DeMarco

Tom DeMarco in 2009

<https://www.infoq.com/news/2009/08/demarc-o-software-engineering-/>

Summary

- Rather than spending time running more and more self-report studies, we should spend time thinking about **study designs beyond self-reports**.
- Software metrics are a complex topic, but we as researchers can and should educate readers to **critically reflect on published results**.

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

Appendix

What Do Companies Measure?

Measuring Developer Productivity: Real-World Examples

A deepdive into developer productivity metrics used by Google, LinkedIn, Peloton, Amplitude, Intercom, Notion, Postman, and 10 other tech companies.



GERGELY OROSZ AND ABI NODA

JAN 16, 2024

Developer Productivity Metrics at 17 Tech Companies

Company	Name of function	What they measure
 Amplitude	Platform Engineering	Ease of Delivery, Engineer Engagement, Deploys per Service, Change Failure Rate
 ATlassian	Developer Experience	DSAT (Developer Satisfaction), Time from Commit to Deploy, Pull Request Cycle Time, Self-Serve Documentation, Self-Serve Dependency Maintenance
 chime	Engineering Operations	Developer CSAT, Perceived Rate of Delivery, Ease of Delivery, Sentiment and Quantitative Metrics for 16+ DevEx factors
 DOORDASH	Developer Platform	Developer Satisfaction, Adoption Rate, Stability of Services/Apps
Etsy	Enablement	Experiment Velocity, Availability, Performance, Developer Sentiment
 GoodRx	Engineering Solutions	Engagement, Weekly Time Loss, Deep Work, Ease of Delivery, Speed, Stability, % Adoption of Standards
Google	Engineering Productivity	Speed, Ease, and Quality *Specific metrics for each of these dimensions vary
 INTERCOM	Developer Experience	Ease of Delivery, Perceived Productivity, Engineer Engagement
 Lattice	Developer Experience	Ease of Delivery, CI Pipeline Failure Rate, CI Pipeline P50 Build Time, Number of Deployments, Change Failure Rate
 LinkedIn	Productivity & Happiness	Developer Build Time, Code Reviewer Response Time, Post-Commit CI Speed, CI Determinism, Deployment Success Rate, Developer NSAT (Net User Satisfaction)
 Microsoft	Engineering Thrive	SPACE metrics *Specific metrics for each of these dimensions vary
 Notion	Developer Infrastructure	Developer survey with a focus on perceptions of friction
PELOTON	Tech Enablement	Developer Satisfaction Score, Time to 1st and 10th PR, Lead Time, Deployment Frequency, % of PRs under 250 lines, Line Coverage, Change Failure Rate, Time to Restore Services
 POSTMAN	Platform Engineering	Ease of Delivery, Perceived Productivity, Weekly Time Loss, Engineer Engagement, Cycle Times, Satisfaction
 Spotify	Developer Experience	Self-Reported Productivity, % Adoption to Golden Standards, Deployment Frequency, Weekly Deployments per Weekly Active Developer
stripe	Developer Tools	Avg. Number of Days With Sufficient Focus Time, Branch Creation to Master Merge Time, Sentiment, Weekly # of PRs per Developer
Uber	Developer Platform	Weekly # of Diffs/PRs per Engineer, Weekly # of Code Reviews per Engineer, Weekly # of Design Docs Generated per Engineer, Avg. Weekly Focus Time per Engineer

getdx.com

pragmaticengineer.com

What Do Companies Measure?

Company	Name of function	What they measure
 Amplitude	Platform Engineering	Ease of Delivery, Engineer Engagement, Deploys per Service, Change Failure Rate
 ATlassian	Developer Experience	DSAT (Developer Satisfaction), Time from Commit to Deploy, Pull Request Cycle Time, Self-Serve Documentation, Self-Serve Dependency Maintenance
 chime®	Engineering Operations	Developer CSAT, Perceived Rate of Delivery, Ease of Delivery, Sentiment and Quantitative Metrics for 16+ DevEx factors
 DOORDASH	Developer Platform	Developer Satisfaction, Adoption Rate, Stability of Services/Apps
 Etsy	Enablement	Experiment Velocity, Availability, Performance, Developer Sentiment
 GoodRx	Engineering Solutions	Engagement, Weekly Time Loss, Deep Work, Ease of Delivery, Speed, Stability, % Adoption of Standards
 Google	Engineering Productivity	Speed, Ease, and Quality *Specific metrics for each of these dimensions vary
 INTERCOM	Developer Experience	Ease of Delivery, Perceived Productivity, Engineer Engagement

 Lattice	Developer Experience	Ease of Delivery, CI Pipeline Failure Rate, CI Pipeline P50 Build Time, Number of Deployments, Change Failure Rate
 LinkedIn	Productivity & Happiness	Developer Build Time, Code Reviewer Response Time, Post-Commit CI Speed, CI Determinism, Deployment Success Rate, Developer NSAT (Net User Satisfaction)
 Microsoft	Engineering Thrive	SPACE metrics *Specific metrics for each of these dimensions vary
 Notion	Developer Infrastructure	Developer survey with a focus on perceptions of friction
 PELOTON	Tech Enablement	Developer Satisfaction Score, Time to 1st and 10th PR, Lead Time, Deployment Frequency, % of PRs under 250 lines, Line Coverage, Change Failure Rate, Time to Restore Services
 POSTMAN	Platform Engineering	Ease of Delivery, Perceived Productivity, Weekly Time Loss, Engineer Engagement, Cycle Times, Satisfaction
 Spotify	Developer Experience	Self-Reported Productivity, % Adoption to Golden Standards, Deployment Frequency, Weekly Deployments per Weekly Active Developer
 stripe	Developer Tools	Avg. Number of Days With Sufficient Focus Time, Branch Creation to Master Merge Time, Sentiment, Weekly # of PRs per Developer
 Uber	Developer Platform	Weekly # of Diff/PRs per Engineer, Weekly # of Code Reviews per Engineer, Weekly # of Design Docs Generated per Engineer, Avg. Weekly Focus Time per Engineer

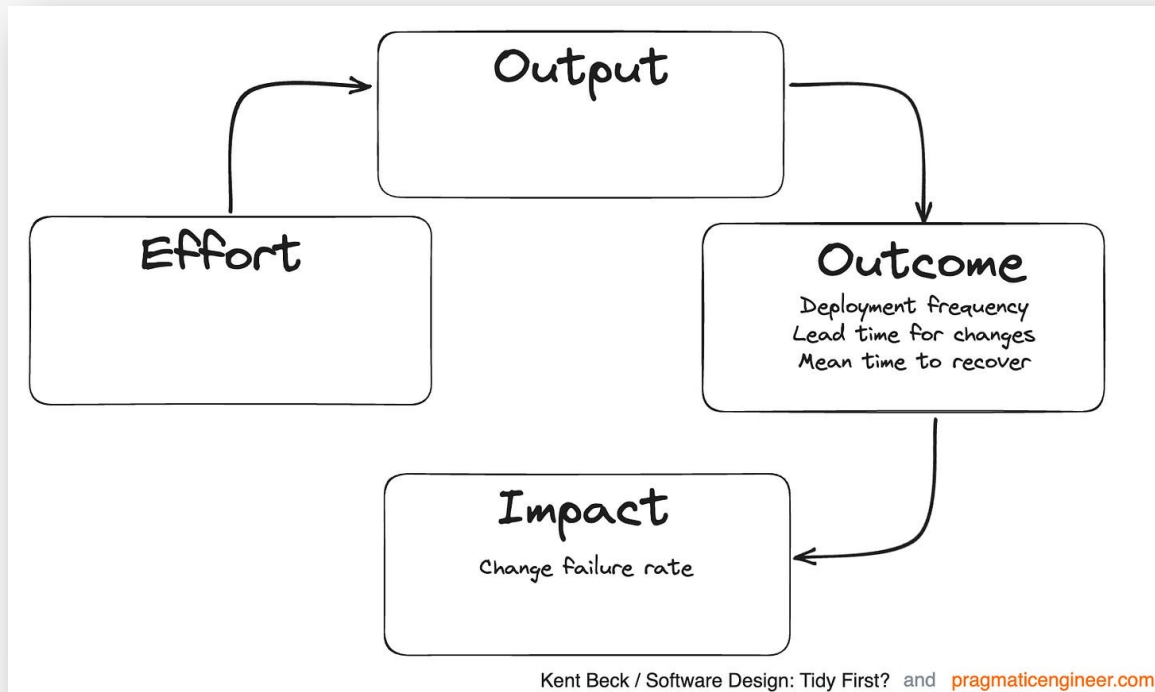
Measuring Developer Productivity: Real-World Examples

A deepdive into developer productivity metrics used by Google, LinkedIn, Peloton, Amplitude, Intercom, Notion, Postman, and 10 other tech companies.



GERGELY OROSZ AND ABI NODA
JAN 16, 2024

DORA Metrics



- Measuring **effort** as a proxy for productivity is usually a bad idea. (e.g., work time, number of commits, etc.)
- Focus on **outcomes** or **impact**.
- DORA metrics do that, however:
 - Lack of scientific **empirical validation** of those metrics.
 - They focus on **very specific** forms of outcomes and impacts.
 - They focus on parts of the process that highly depend on **(automated) tooling**.

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

DORA, SPACE, and DevEx

DORA

4 key metrics
published in 2020:

Deployment frequency

Lead time for changes

Change failure rate

Time to restore service

SPACE

5 dimensions
published in 2021:

S atisfaction and wellbeing

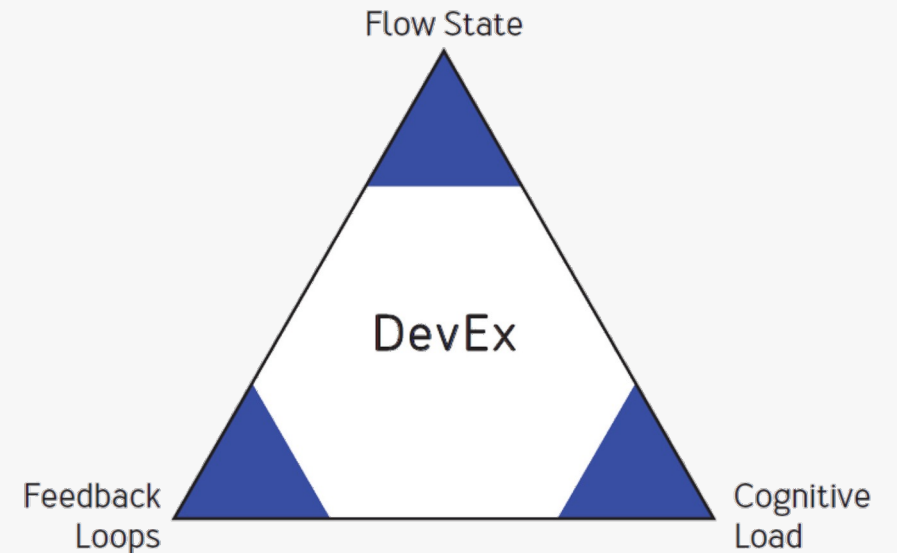
P erformance

A ctivity

C ommunication and collaboration

E fficiency and flow

pragmaticengineer.com



<https://newsletter.pragmaticengineer.com/p/developer-productivity-a-new-framework>

DORA, SPACE, and DevEx

TABLE 1: **EXAMPLE DEVEX METRICS**

	FEEDBACK LOOPS	COGNITIVE LOAD	FLOW STATE
PERCEPTIONS <i>Human attitudes and opinions</i>	• Satisfaction with automated test speed and output • Satisfaction with time it takes to validate a local change • Satisfaction with time it takes to deploy a change to production	• Perceived complexity of codebase • Ease of debugging production systems • Ease of understanding documentation	• Perceived ability to focus and avoid interruptions • Satisfaction with clarity of task or project goals • Perceived disruptiveness of being on-call
WORKFLOWS <i>System and process behaviors</i>	• Time it takes to generate CI results • Code review turnaround time • Deployment lead time (time it takes to get a change released to production)	• Time it takes to get answers to technical questions • Manual steps required to deploy a change • Frequency of documentation improvements	• Number of blocks of time without meetings or interruptions • Frequency of unplanned tasks or requests • Frequency of incidents requiring team attention
KPIS <i>North star metrics</i>	• Overall perceived ease of delivering software • Employee engagement or satisfaction • Perceived productivity		

<https://dl.acm.org/doi/10.1145/3595878>

The GitClear Report

Code Change Definitions

To analyze how code quality is changing, we will review the differences in types of code changes observed in 2023 vs. the years prior, when AI was much less prevalent. [GitClear classifies code changes](#) (operations) into seven categories. The first six operations are analyzed in this research:

1. **Added code.** Newly committed lines of code that are distinct, excluding lines that incrementally change an existing line (labeled "Updates"). "Added code" also does not include lines that are added, removed, and then re-added (these lines are labeled as "Updated" and "Churned")
2. **Deleted code.** Lines of code that are removed, committed, and not subsequently re-added for at least the next two weeks.
3. **Moved code.** A line of code that is cut and pasted to a new file, or a new function within the same file. By definition, the content of a "Moved" operation doesn't change within a commit, except for (potentially) the white space that precedes the content.
4. **Updated code.** A committed line of code based off an existing line of code, that modifies the existing line of code by approximately three words or less.
5. **Find/Replaced code.** A pattern of code change where the same string is removed from 3+ locations and substituted with consistent replacement content.
6. **Copy/Pasted code.** Identical line contents, excluding programming language keywords (e.g., `end`, `}`), `[]`, that are committed to multiple files or functions within a commit.
7. **No-op code.** Trivial code changes, such as changes to white space, or changes in line number within the same code block. No-op code is excluded from this research.

	Added	Deleted	Updated	Moved	Copy/pasted	Find/replaced	Churn
2020	39.2%	19.5%	5.2%	25.0%	8.3%	2.9%	3.3%
2021	39.5%	19.0%	5.0%	24.8%	8.4%	3.4%	3.6%
2022	41.0%	20.2%	5.2%	20.5%	9.4%	3.7%	4.0%
2023	42.3%	21.1%	5.5%	16.9%	10.5%	3.6%	5.5%
2024	43.6%	22.1%	5.8%	13.4%	11.6%	3.6%	7.1%

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