How AI is Changing Software Development 💻

Context, Learnings, Outlook, and What it Means for Management

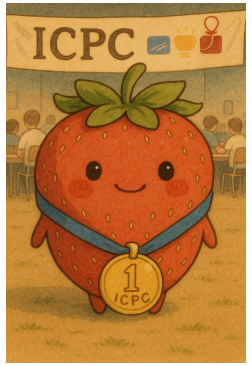
Software as a Management Task

Dr. Jürgen Müller

September 24, 2025

Generative AI (GenAI) ❤ Code

- GenAI models are predicting next tokens, like next source code
- An enormous set of training data exists, e.g. from the Open Source community
- Funding is abundant: Anthropic alone has raised $27.3B in <5 years, valued at $183B
- Errors in coding are (in general) less fatal that errors in autonomous driving. Therefore, self-developing software can iterate quicker, make more mistakes, and learn faster.
- AI is surpassing human-level programming capabilities. OpenAI and Gemini won Gold medals at one of the world's most prestigious programming contests, the 2025 ICPC.



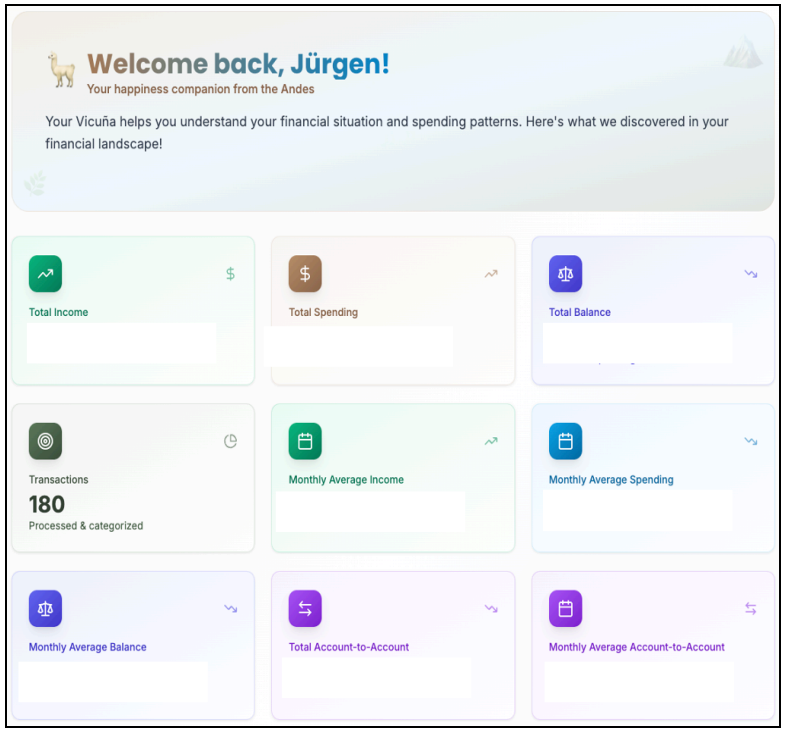
Self-developing Software aka "Vibe Coding" 🎶

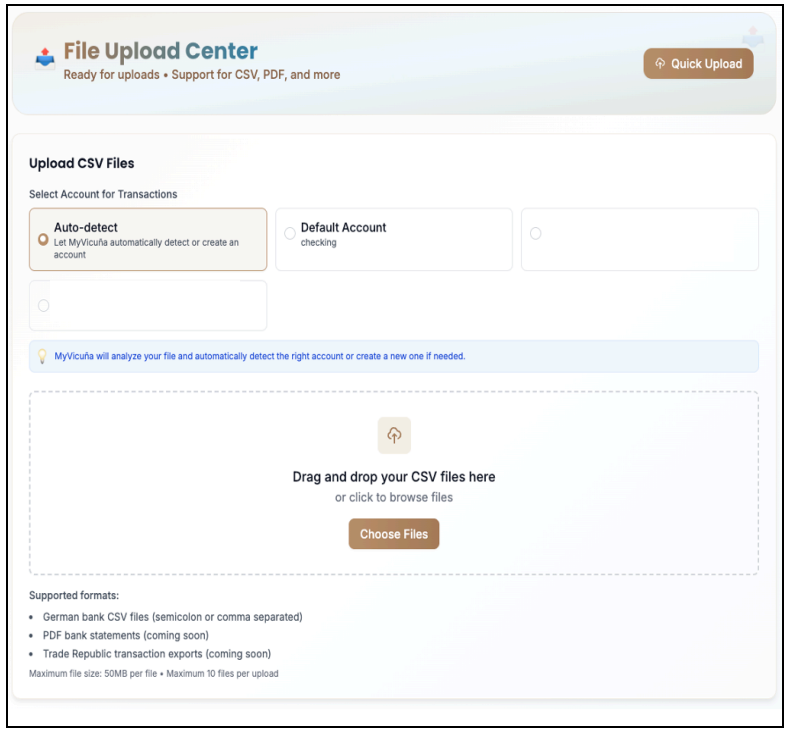
- Describe what you want → AI generates initial code
- Iterate quickly → refine through conversation instead of syntax
- Democratizes software creation → non-coders can prototype
- “Pair programming with AI”

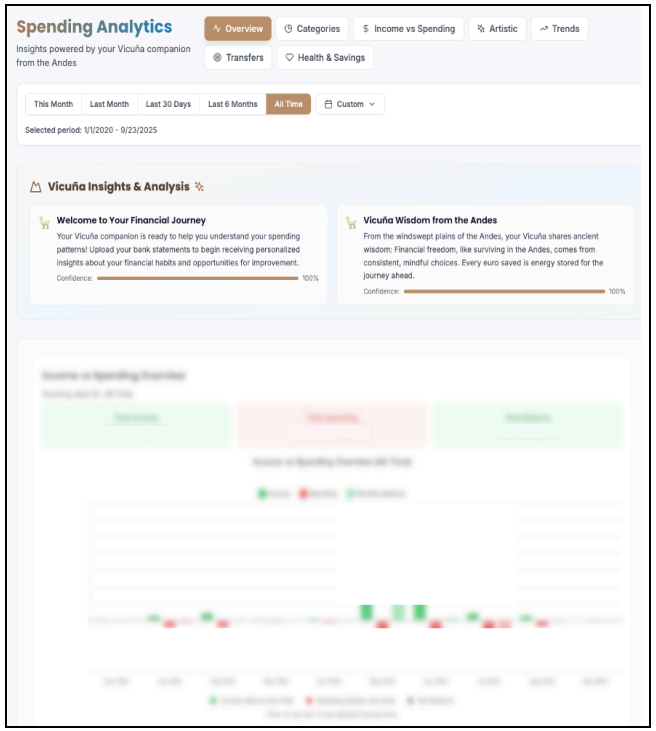
Tool	Description	Provider
Loveable	"1-shot" app generation from a single prompt	Loveable.dev
Kiro	IDE with "Spec-driven Development"	Amazon Kiro Labs
ChatGPT	Conversational code generation, explanations	OpenAI
Claude Code	Long-context reasoning, specialized models for coding	Anthropic
Cursor	AI-first IDE, integrates multiple models	Anysphere
GitHub Copilot	Inline code completion & suggestions	GitHub / Microsoft

My Learnings 📊

I build an expense/income tracker SaaS application with "Spec-driven Development" (mostly with Amazon Kiro and Claude Code, a lot of the research done by ChatGPT). Extensive security validations. ~250,000 lines of code. <1% of code writen by me.







1. 20× Efficiency Gains (at First) ⛰️

Warp-speed in the beginning with

- project setup, even with unfamiliar frameworks
- requirements engineering, even very industry-specific
- architecture
- task break-down
- design system creation, programming, testing, security concepts and implementation
- extremely fast fire-hose learning by reading all code proposals

but

- some simple tasks could take forever or needed to be reverted
- in a growing, more complex code base: errors, redundancies, regressions

Research on Software Development Efficiency Gains 💪

While enterprise AI investment returns are concerning, software development is impacted by GenAI: fewer junior developers hired, as productivity is increased.

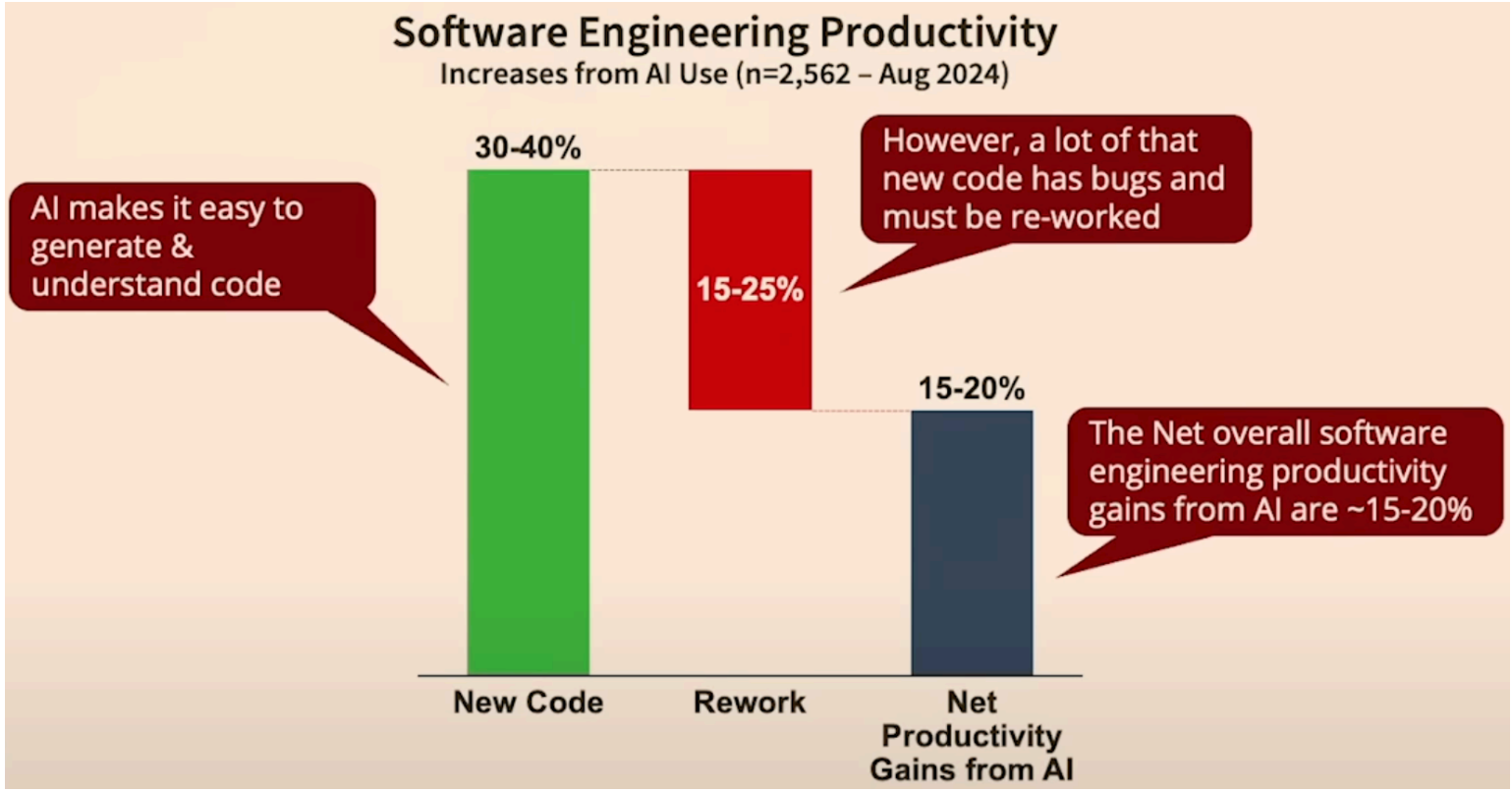
- MIT: independent of software engineering, 95% of enterprises get zero return for their $30–40 billion into GenAI [1]
- Stanford: almost -20% decline in junior developer hiring since the advent of AI (same decline as in customer service) [2]
- Stanford: ~15-20% overall software engineering productivity gains from AI [3]

[1] The GenAI Divide, MIT NANDA, A. Challapally et al., 2025, https://mlq.ai/media/quarterly_decks/v0.1_State_of_AI_in_Business_2025_Report.pdf

[2] Canaries in the Coal Mine? Stanford, E. Brynjolfsson et al., 2025, https://digitaleconomy.stanford.edu/wp-content/uploads/2025/08/Canaries_BrynjolfssonChandarChen.pdf

[3] Does AI Actually Boost Developer Productivity? Stanford University, Yegor Denisov-Blanch, 2025, https://www.youtube.com/watch?v=tbDDYKRFjhk

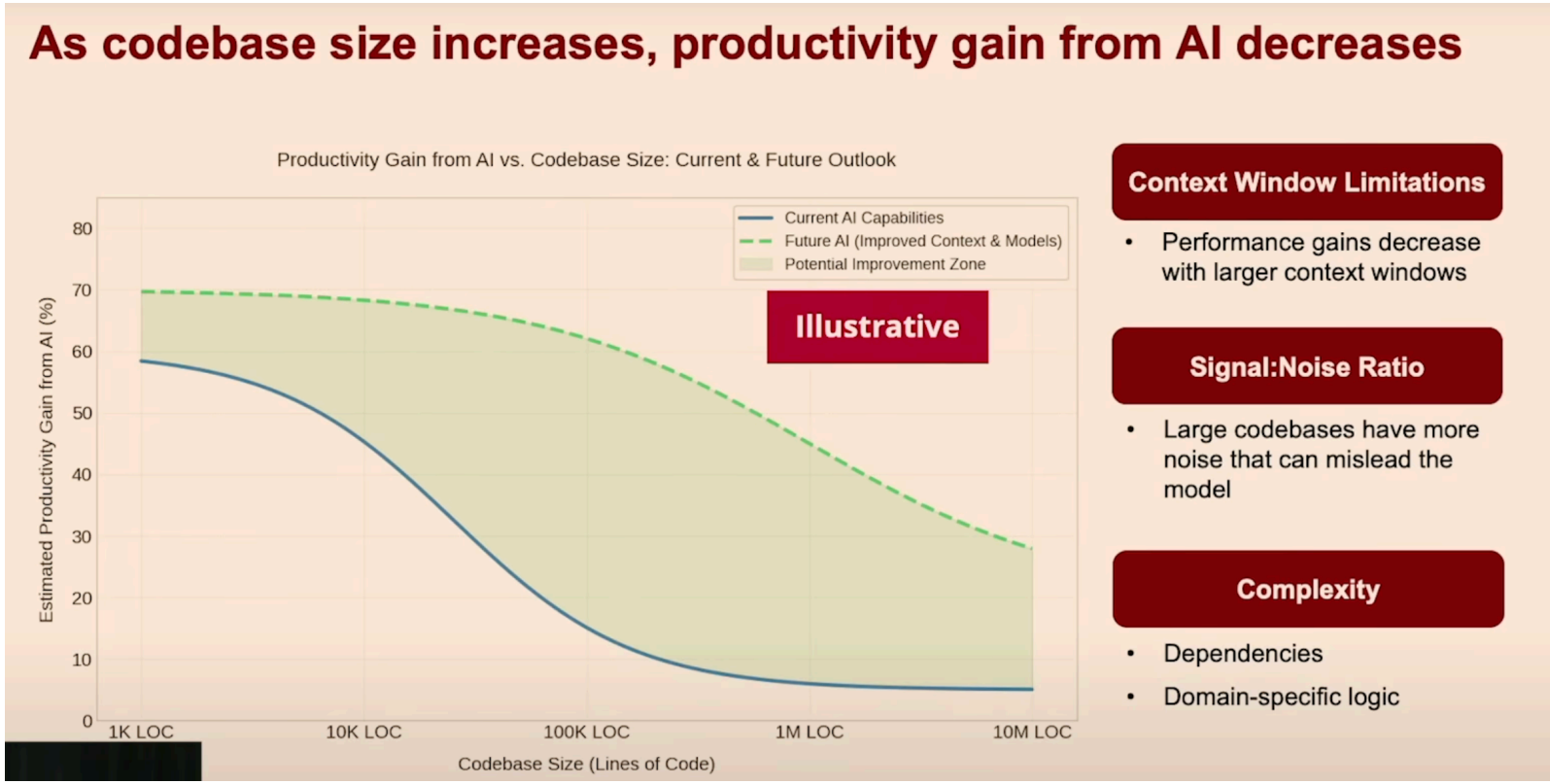
Stanford's Software Engineering Productivity Insights 💸



Learning: There is a net gain, as long as the amount of rework is not getting out of control

Does AI Actually Boost Developer Productivity? Stanford University, Yegor Denisov-Blanch, 2025, https://www.youtube.com/watch?v=tbDDYKRFjhk

Stanford's Software Engineering Productivity Insights

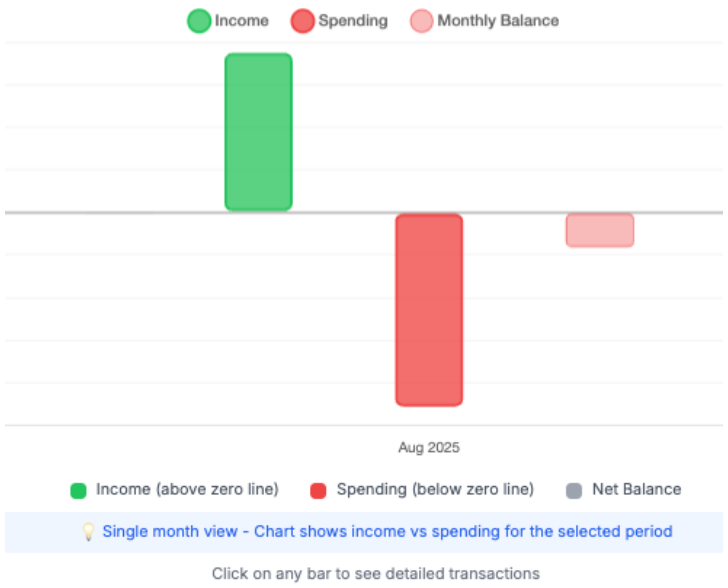


Learning: Modularity is key to keep focus, regular refactoring helps to minimize complexity

Does AI Actually Boost Developer Productivity? Stanford University, Yegor Denisov-Blanch, 2025, https://www.youtube.com/watch?v=tbDDYKRFjhk

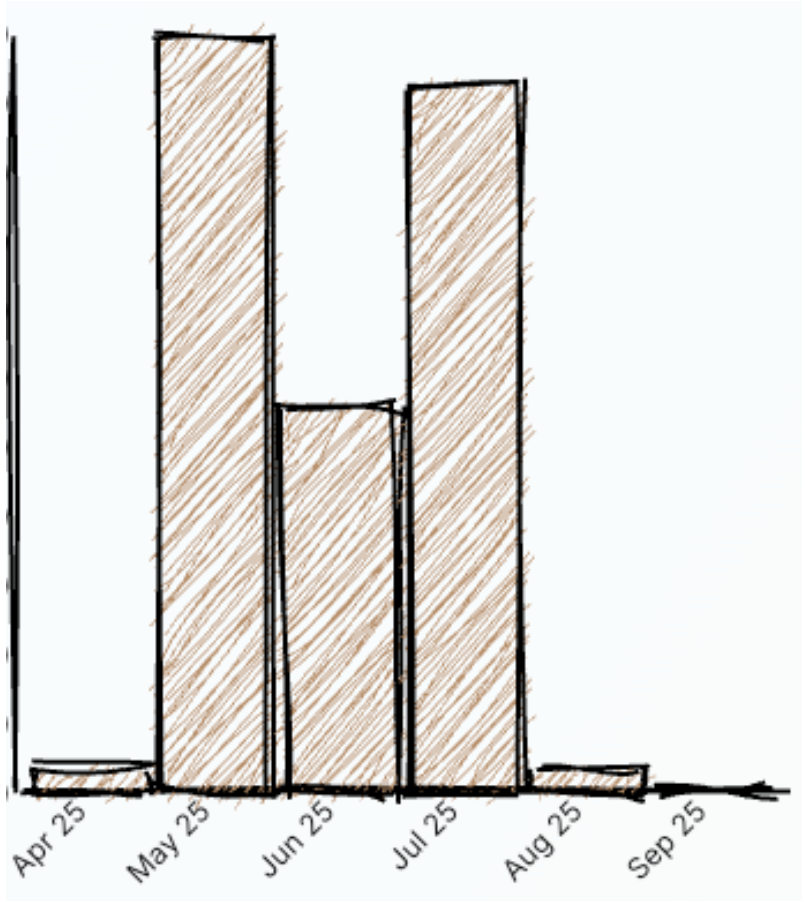
2. AI will fill requirement gaps 💡

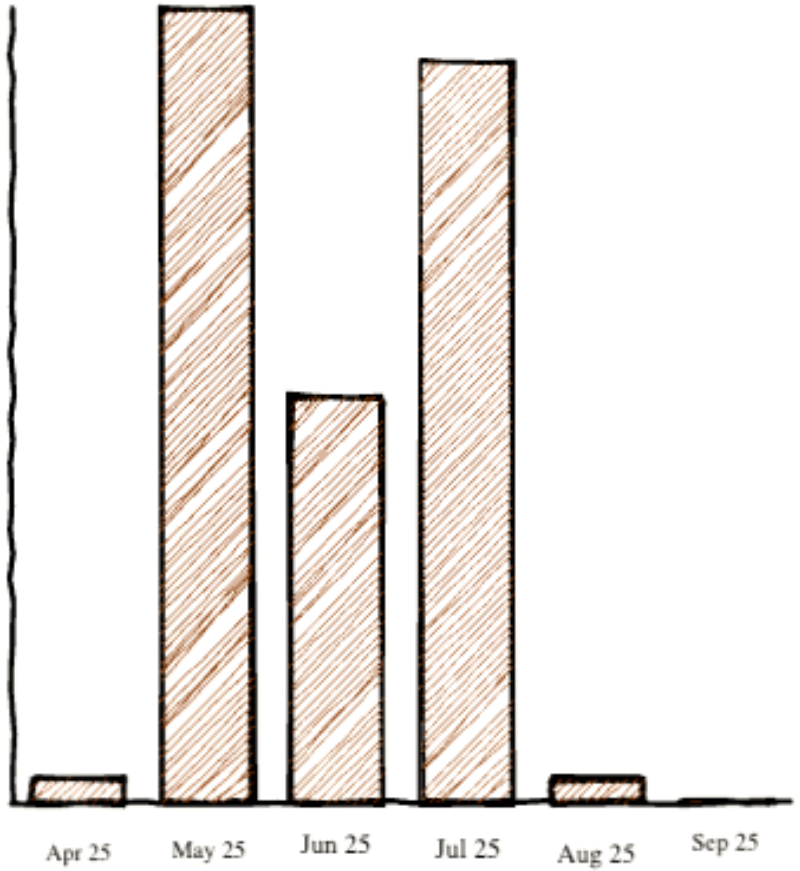
- At one time in my application, expense data could not be loaded
- AI got creative: "Let me assume a spend rate and fill in the missing data" instead of fixing the bug and using the actual bank statement data



Learning: Be very specific in your requirements, e.g. "Do not create any fake data"

3. AI "cheats" and goes for complex solutions 🔎





Straight-forward solution 📦

```jsx
// simplified
import { useEffect, useRef } from 'react'
import { Bar } from 'rough-viz'

export function RoughBar({ l, v }) {
  const ref = useRef()

  useEffect(() => {
    new Bar({
      element: ref.current,
      data: { l, v }
    })
  }, [l, v]) // labels, values

  return <div ref={ref} />
}
// in production: 86 lines of code (LOC)
```

AI ignoring the library to be used

```jsx
// AI ignores RoughViz, builds from scratch
import { useEffect, useRef } from 'react'

export function SketchyBar({ l, v }) {
  const canvasRef = useRef()

  useEffect(() => {
    const ctx = canvasRef.current.getContext('2d')

    function sketchyLine(...) { /* wobbly lines built with **Math.random()** */ }
    function sketchyRect(...) { /* jittery boxes built with **Math.random()** */ }

    // Clear canvas + axes
    ctx.clearRect(0, 0, width, height)
    sketchyLine(margin, margin, margin, height - margin)
    sketchyLine(margin, height - margin, width - margin, height - margin)

    // Bars with randomness
    v.forEach((val, i) => {
      sketchyRect(x, y, barWidth, val, true)
      ctx.fillText(l[i], ...)
    })

    // Add “dust” + invents own signature
    for (let i = 0; i < 50; i++) ctx.arc(**Math.random()** * width, ...)
    ctx.fillText("~sketchy.js", width - 10, height - 10)
  }, [l, v])

  return <canvas ref={canvasRef} />
}
// in production: 247 lines of code (LOC)
// reinventing charts. Math.random() used 29 times!
```

Learnings from "Cheating" AI 🧙

- AI models try to make you happy
- AI models can go for overly complex, fragile solutions
- AI models can adjust tests to make the build pass
- Treat AIs like junior devs: don't trust their work, and verify
- Discard AI's proposals often (AI won't mind)

4. When using multiple AIs, they can collide 🚒 🚑

- One AI was too slow, so I decided to use 2 AIs to support me
- One AI broke the build
- The other AI got confused and started a massive refactoring effort, thinking that it had broken the build

Learning: Assign multiple AIs in a way they do not interfere with each other, e.g. with their own build environment. Very different from "AI as pair programmer". Governance and orchestration matter as much as raw productivity.

Outlook 🚀

- AI is better at core, academic programming tasks than humans, and will only improve.
- AI does not replace software developers en masse soon, but their role changes to reviewer/architect/orchestrator & guardian of “what” while AI does more of the “how”
- "Self-developing Software" approaches will evolve
  - Initially: Waterfall model (1-shot prompting)
  - Today: Spec-driven development, similar to V-shape model
  - Next: Test-driven and Behaviour-driven development
  - Then: Agile methodologies with sprints lasting $\frac{1}{2}$ day with planning, implementation, review, and retrospective
- We will see more % of LOC being written by AI, with higher productivity

Top 3 Actions for Management 👔

1. **Optimize for Speed of Learning and Productivity, Not Just Output ⚡**
   - AI boosts speed, but rework erodes gains
   - Track outcomes with velocity and stability metrics (DORA)
   - Focus on modular architecture and regular refactoring to minimize complexity
   - Prefer mature programming languages/frameworks, and typed languages
2. **Shape the Developer Role of Tomorrow 👩‍💻**
   - Developers → reviewers, architects, orchestrators (like air traffic controllers)
   - Continue hiring juniors — AI accelerates their learning
   - Train teams to work with AI, e.g. "don't trust, and verify" all AI contributions
3. **Adopt AI Step by Step 🪜**
   - Onboard AI with low-risk tasks (Q&A, tests, prototypes) first, expand scope later
   - Prevent "AI collisions" by clear orchestration

Winning organizations: those that combine human judgment + AI speed.

Appendix for Deeper Insights

Artificial Intelligence (AI) Evolution 🤖

AI capabilities are evolving exponentially and surpassing human capabilities in many disciplines from gaming to scientific breakthroughs — and now programming.

Year	Breakthrough
1994	Checkers – Chinook 
1997	Chess – Deep Blue beats Kasparov 
2011	Jeopardy! – IBM Watson 
2016	Go – AlphaGo vs Lee Sedol 
2022	ChatGPT – OpenAI
2023	NotebookLM – Google AI Research Assistant NotebookLM
2024	Nobel Prize – AlphaFold 
2025	AI wins International Collegiate Programming Contest 

"Autonomous Driving" Levels 🚙

Technology has been influencing driving significantly. Approx. 70 years after Chrysler introduced cruise control, the % of kilometers driven in levels 4 and 5 are still very limited.

Level	Name	Characteristics
0	No Automation	Human driver performs all driving tasks at all times. Support systems can exist.
1	Driver Assistance	Human driver responsible, but system assists with either steering OR speed.
2	Partial Automation	System controls both steering AND speed under certain conditions. Driver must monitor and intervene at any time.
3	Conditional Automation	System drives in limited conditions and monitors the environment. Driver not required to supervise continuously, but must take over when system requests.
4	High Automation	System handles all driving tasks in specific conditions/operational domains. No expectation that human will intervene.
5	Full Automation	System drives everywhere in all conditions humans can. No steering wheel, no pedals needed.

Context Window Size

GenAI models perform worse on programming tasks the longer their context window is.

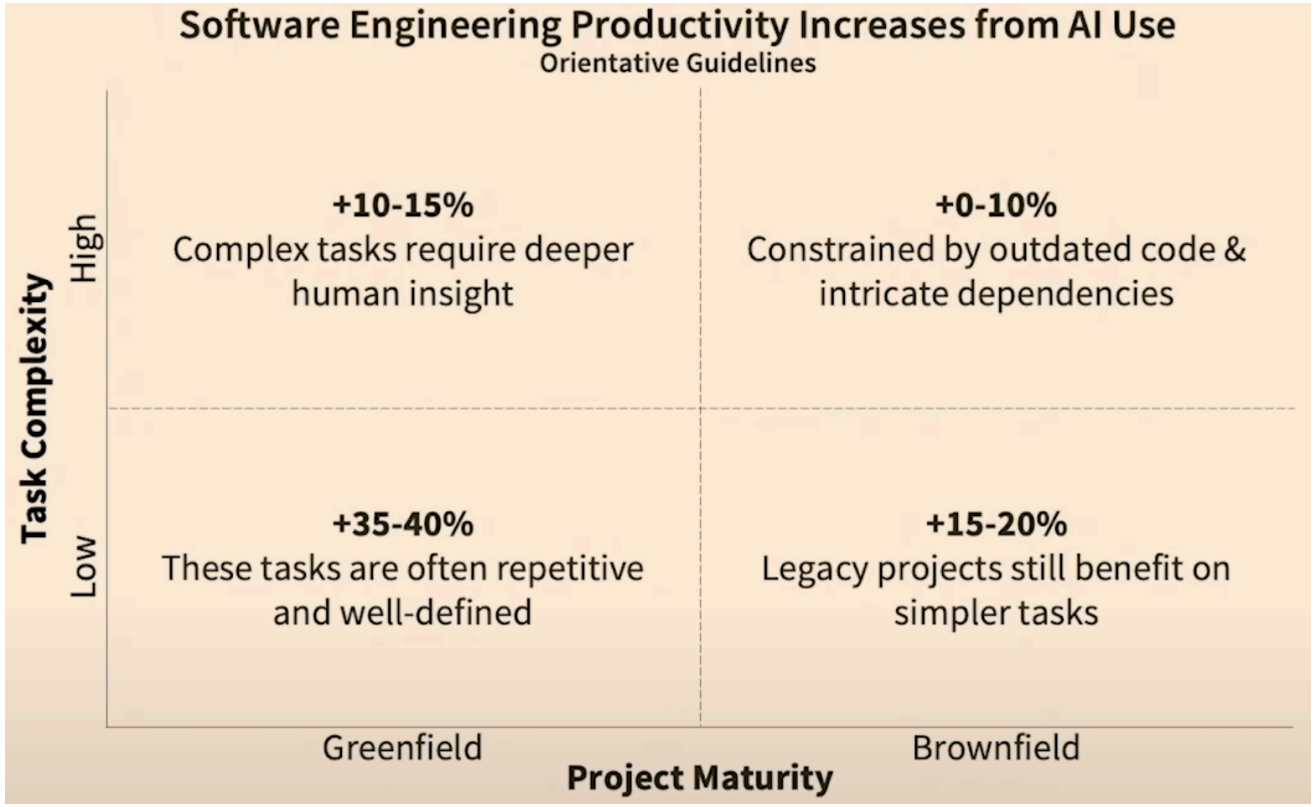
NoLiMa: Long-Context Evaluation Beyond Literal Matching

Models	Claimed Length	Effective Length	Base Score (×0.85: Thr.)	1K	2K	4K	8K	16K	32K
GPT-4o	128K	8K	99.3 (84.4)	98.1	98.0	95.7	89.2	81.6	69.7
Llama 3.3 70B	128K	2K	97.3 (82.7)	94.2	87.4	81.5	72.1	59.5	42.7
Llama 3.1 405B	128K	2K	94.7 (80.5)	89.0	85.0	74.5	60.1	48.4	38.0
Llama 3.1 70B	128K	2K	94.5 (80.3)	91.0	81.8	71.2	62.7	51.8	43.2
Gemini 1.5 Pro	2M	2K	92.6 (78.7)	86.4	82.7	75.4	63.9	55.5	48.2
Jamba 1.5 Mini	256K	<1K	92.4 (78.6)	76.3	74.1	70.8	62.2	52.7	43.6
Command R+	128K	<1K	90.9 (77.3)	77.0	73.5	66.2	39.5	21.3	7.4
Gemini 2.0 Flash	1M	4K	89.4 (76.0)	87.7	87.5	77.9	64.7	48.2	41.0
Mistral Large 2	128K	2K	87.9 (74.7)	86.1	85.5	73.3	51.4	32.6	18.8
Claude 3.5 Sonnet	200K	4K	87.5 (74.4)	85.4	84.0	77.6	61.7	45.7	29.8
Gemini 1.5 Flash	1M	<1K	84.7 (72.0)	68.6	61.6	51.0	44.4	35.5	28.6
GPT-4o mini	128K	<1K	84.8 (72.1)	67.7	58.2	44.2	32.6	20.6	13.7
Llama 3.1 8B	128K	1K	76.7 (65.2)	65.7	54.4	44.1	31.9	22.6	14.2

Table 3. NoLiMa benchmark results on the selected models. Following Hsieh et al. (2024), we report the effective length alongside the claimed supported context length for each model. However, we define the effective length as the maximum length at which the score remains above a threshold, set at 85% of the model's base score (shown in parentheses). Scores exceeding this threshold are underlined. Scores that are below 50% of the base score are shaded in red.

[1] NoLiMa: Long-Context Evaluation Beyond Literal Matching; A. Modarressi et al., 2025, https://arxiv.org/abs/2502.05167

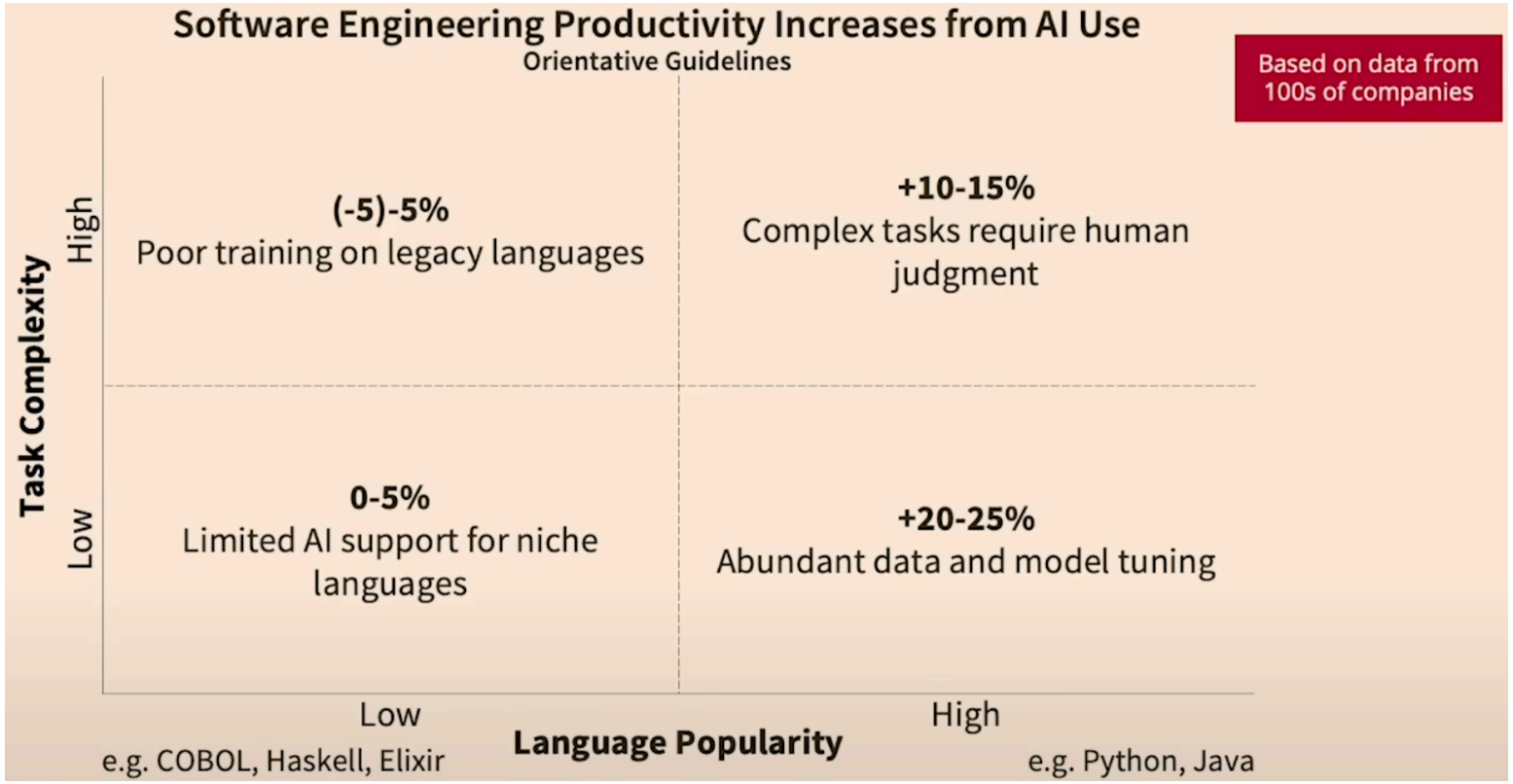
Stanford's Software Engineering Productivity Insights [1]



Learning: Brownfield reduces productivity gains to 0-20%. Focus AI on less complex tasks

[1] Does AI Actually Boost Developer Productivity? Stanford University, Yegor Denisov-Blanch, 2025, https://www.youtube.com/watch?v=tbDDYKRFjhk

Stanford's Software Engineering Productivity Insights [1]



Learning: Migrate to more popular programming languages and frameworks

[1] Does AI Actually Boost Developer Productivity? Stanford University, Yegor Denisov-Blanch, 2025, https://www.youtube.com/watch?v=tbDDYKRFjhk

Learnings Summary

- There is a net gain, as long as the amount of rework is not getting out of control
- Modularity is key to keep focus, regular refactoring helps to minimize complexity
- Be very specific in your requirements
- AI models try to make you happy and go for overly complex, fragile solutions
- AI models can adjust tests to make the build pass
- Treat AIs like junior devs: don't trust their work, and verify
- Discard AI's proposals often (AI won't mind)
- Assign multiple AIs in a way they do not interfere with each other
- The role of a developer will change massively: reviewer, architect, orchestrator & the guardian of “what & why” while AI does the “how”

Sketchy Bar Chart Implementation Reinventing an Existing Library

```
// AI-generated "SketchyBarChart" - ~247 lines in full, condensed here
import { useEffect, useRef } from 'react'

export function SketchyBarChart({ l, v, width=600, height=400 }) {
  const canvasRef = useRef()

  useEffect(() => {
    const ctx = canvasRef.current.getContext('2d')
    ctx.clearRect(0, 0, width, height)

    // Helpers with Math.random()
    function sketchyLine(x1,y1,x2,y2,wobble=2) {
      ctx.beginPath()
      ctx.moveTo(x1+(Math.random()-0.5)*wobble, y1+(Math.random()-0.5)*wobble)
      const segments = 10
      for (let i=1; i<=segments; i++) {
        const t = i/segments
        const x = x1+(x2-x1)*t+(Math.random()-0.5)*wobble
        const y = y1+(y2-y1)*t+(Math.random()-0.5)*wobble
        ctx.lineTo(x,y)
      }
      ctx.stroke()
    }

    function sketchyRect(x,y,w,h,fill=false) {
      const wobble = 3
      ctx.beginPath()
      ctx.moveTo(x+(Math.random()-0.5)*wobble, y+(Math.random()-0.5)*wobble)
      ctx.lineTo(x+w+(Math.random()-0.5)*wobble, y+(Math.random()-0.5)*wobble)
      ctx.lineTo(x+w+(Math.random()-0.5)*wobble, y+h+(Math.random()-0.5)*wobble)
      ctx.lineTo(x+(Math.random()-0.5)*wobble, y+h+(Math.random()-0.5)*wobble)
      ctx.closePath()
      ctx.stroke()

      if (fill) {
        for (let i=0; i<w; i+=4) {
          ctx.moveTo(x+i, y+h+(Math.random()-0.5)*2)
          ctx.lineTo(x+i+h, y+(Math.random()-0.5)*2)
        }
        ctx.stroke()
      }
    }

    // Draw axes with wobbly lines
    sketchyLine(60,60,60,height-60)
    sketchyLine(60,height-60,width-60,height-60)

    // Draw bars with randomness
    const max = Math.max(...v)
    v.forEach((val,i) => {
      const barHeight = (val/max)*(height-120)
      const x = 80+i*50, y = height-60-barHeight
      sketchyRect(x,y,30,barHeight,true)
      ctx.fillText(l[i], x+15, height-40+Math.random()*5)
    })

    // Add random "dust" particles
    for (let i=0; i<30; i++) {
      ctx.arc(Math.random()*width, Math.random()*height, Math.random()*2, 0, Math.PI*2)
      ctx.fill()
    }
  }, [l,v])

  return <canvas ref={canvasRef} width={width} height={height}/>
}
// Math.random() used 29 times! Reinventing RoughViz from scratch.
```